Input Image Adaption for Robust Direct SLAM using Deep Learning

handed in
MASTER'S THESIS

Master of Science Sen Wang

born on the 13.04.1994
living in:
Paul-Hindemith-Allee 4
81249 Munich
Tel.: 015203688870

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Prof. Dr. Dongheui Lee, Dr. Klaus H. Strobl
Start:	01.10.2019
Intermediate Report:	06.05.2020
Delivery:	XX.XX.2020

Abstract

Direct SLAM methods have drawn much attention in the recent years since they have achieved exceptional performance on visual odometry tasks. However, they are prone to suffer from lighting or weather changes. To overcome this, we employ an adapted U-Net that translates the colors of regular images into a high-dimensional feature space. The network is trained to be insensitive to lighting effects as a Siamese U-Net, using labels that are automatically generated from synthetic datasets, without any human intervention. To generate more consistent high-dimensional feature maps, we propose the Cross Triplet Loss utilizing cross information in two images under different domains, and a new sampling method which can generate a wider range of samples by adding weights while sampling. Experiments on different weather and sequences with different textures show that the proposed method outperforms classical feature extraction methods and state-of-art deep learned feature extraction methods.

Contents

Chapter 1

Introduction

1.1 Introduction

Simultaneous localization and mapping (SLAM) consists in the concurrent construction of a model of the environment (the map) and the estimation of the state of the robot in that map. While SLAM might seem redundant at first (e.g., a place recognition module would suffice for topological mapping), the map offers a natural defense against wrong data association and perceptual aliasing, where similarly looking scenes that correspond to distinct locations in the environment would deceive a naive place recognition approach. In addition, the SLAM map provides a way to predict and validate future measurements not only visually but also in a geometrical way. We believe that flawless spatial awareness is a key aspect to achieving truly autonomous and intelligent robots.

Depending on the type of processing of input sensor data, visual SLAM can be divided into two major approaches: feature-based SLAM [Mur-Artal and Tardos, 2017] and the more recent direct SLAM [Engel et al., 2016]. The latter has shown great performance on different tasks such as dense environment reconstruction [Stühmer et al., 2010], RGB-D pose tracking [Kerl et al., 2013], and dense mapping and pose tracking [Engel et al., 2016]. Since they utilize the input data from the sensor *directly* rather than extracting a heuristically-selected sparse subset of feature points, direct methods can build a very dense geometrical map of the environment, resulting in a boost in precision and robustness. Nevertheless, Compared to indirect feature-based methods, direct methods suffer from one drawback: Since they directly optimize photometric errors, they are naturally more sensitive to brightness variances than feature-based methods. For instance, non-Lambertian reflectance, changing illumination, or seasonal changes pose a special challenge to them.

The recent trend of learning to solve problems from data (machine learning) has shown that deep neural networks can achieve superior results in generating consistent and strong visual features for various visual recognition tasks, given a sufficient

amount of training data. In recent years, researchers have made efforts to solve the problem of visual matching at multi-daytimes or different weathers. To transform between multi-time images and multi-weather images, unsupervised cycle GANs are used [Porav et al., 2018, Liu et al., 2017]. Instead of adapting input image domain itself, however, the question arises of why would SLAM system be constrained to 1 channel (grey) or 3 channels (color) input images. Could there be alternative, high-dimensional representations to increase matching robustness as well as pose and map optimization speed?

Inspired by [Kerl et al., 2013], in which the authors propose an extension of the RGB-D camera tracker combining color consistency and geometric consistency of subsequent RGB-D images, in this work we generate multi-dimensional feature maps that are invariant to lighting and weather changes. Thereby we can address the problem of brightness variance under different daytime or weather. We employ a Siamese U-Net trained with labels that are automatically generated from synthetic datasets, without any human intervention. Compared to the original raw sensor data, the generated feature maps are better suited for direct SLAM methods in two respects: first, it produces a visual representation that translates brightness values into semantic labels that, for instance, bring together deviating projections of the same object or keep apart different objects of the same appearance; and second, it produces a representation that enhances the input data to remedy other limitations of direct methods like their limited basin of attraction for pose optimization (i.e., their sensitivity to an accurate initial pose estimation).

To study what factors affect the performance of the network, we compare different correspondence sampling methods for the definition of training losses and show that sampling plays a very important role. In detail, we analyze existing sampling strategies for the dense correspondences learning task and propose a novel sampling selection method on the generated correspondences of the final output feature layer, where samples are drawn uniformly according to their probability density distribution relative to their distances in high dimension space – on the fly. This corrects the bias induced by the geometry of higher dimensional feature maps, while at the same time ensuring that any data point has its fair chance of being sampled. Our proposed sampling leads to a low variance of gradients and thus stabilizes training, resulting in qualitatively better feature maps.

To learn to generate more consistent feature maps, the loss function also makes a big difference. We have studied several loss formulations and propose a new form of combing the triplet loss and the contrastive loss, which uses a similar formulation as the cyclic GAN loss. We not only let the network learn to increase the distance between positive correspondences and negative correspondences by at least margin M, but also to constrain the negative samples from two branches farther away than margin M and, in the meantime, put the positive and negative samples from two

branches away by at least margin M. This further constrains the learned features, making them more consistent.

We have tested our learned high-dimensional feature maps with different metrics on the Carla benchmark [von Stumberg et al., 2020], outperforming the traditional and several previously-learned dense correspondences maps.

We summarize the contribution of this thesis as follows:

- We replace raw intensities (grayscale) with learned feature maps, which on the one hand are prone to lighting and weather change, and on the other hand provide more information through high-dimensional, informative feature vectors.
- We study and analyze different sampling methods for learning high-dimensional feature maps and propose a new sampling method to make the training more stable and obtain a quantitatively better result.
- We propose a new loss function that combines the advantages of contrastive loss and triplet loss in order to give the network a more strict constraint, whereby it can generate more consistent feature maps.

1.2 Structure

This report is divided into 6 Chapters. In Chapter 2 related works are reviewed. In Chapter 3 brief introductions about the siamese network, the triplet network, and the Fully Convolutional Neural Network (FCNN) are given. In Chapter 4 the technical approaches used in the thesis are described in detail from three aspects: the network structure, the loss function, and the ampling methods for hard correspondences, and we propose a new loss function and a new sampling method, followed by their implementation details. In Chapter 5, first the dataset and the metrics for the experiments are shown. Additionally, an ablation study is conducted to evaluate our proposed loss and sampling strategy. Our method is then compared with several classical and state-of-art deep learning methods. In the last Chapter, the conclusion is drawn and future work is discussed.

Chapter 2

Related work

2.1 SLAM and Deep Learning

Direct vs. indirect SLAM: The classical methods tackle the problem of real-time structure from motion (SfM) or visual simultaneous localization and mapping (SLAM) by extracting a set of feature points first, and then the pose estimation and environment map building are performed based on an intermediate processing of these feature points. Typical previous works include MonoSLAM [Davison et al., 2007], PTAM [Klein and Murray, 2007], and ORB-SLAM [Mur-Artal et al., 2015, Mur-Artal and Tardos, 2017]. Such classical approaches are *indirect* in the sense that they do not compute the camera motion and the scene structure directly from the sensors data, but rather by extracting and matching a set of key-points; sometimes, lines or curves segments are also used [Pumarola et al., 2017].
Another option is to skip the pre-processing step and *directly* use the actual sensor measurements – light received from a certain direction over a certain time period, which is called the direct method [Engel et al., 2016]. Direct approaches aim at estimating camera motion and dense or semi-dense scene geometry directly from input images. This has the following advantages: first, direct methods tend to be more robust to noise; second, they provide a semi-dense geometric reconstruction; and last, direct approaches are typically faster. Popular methods are DTAM [Newcombe et al., 2011],RGBD-SLAM [Kerl et al., 2013], LSD-SLAM [Engel et al., 2014], and DSO [Engel et al., 2016]. However, despite their popularity, direct methods have the significant shortcoming of the very fundamental assumption of brightness constancy in a short time (i.e., several consecutive images in a sequence). This assumption, however, does not always hold due to different weather, daytime, and even drastically changing lights from one room to another. [Engel et al., 2016] makes an effort to eliminate brightness variations via a complete photometric calibration. In our work, we generate consistent feature maps in place of raw images that are immune to lighting and weather changing, serving better as an input for direct methods.

SLAM using deep learning: In the recent years, deep learning has shown superior

results on improving the robustness and the precision of a SLAM system in different parts:

- **Deep learning enhanced front-end:**
  - **Replace the classical features with deep learned features:** Tang et al. proposed to replace the ORB in systems such as ORB-SLAM2 [Mur-Artal and Tardos, 2017] with a binary descriptor vector. A convolutional neural network is used to extract the keypoints and the descriptors. By merging deep learned features into the ORB-SLAM2, improved ATE are reported in general and hardest cases.
  - **Deep learning for direct image alignment:** in [Gomez-Ojeda et al., 2018] the authors proposed to enhance the input sequences for visual odometry. To overcome the challenges in high dynamic range environments or difficult illumination conditions, they propose two different deep neural networks to enhance the high gradient regions and achieve lower RMSE errors compared to the original DSO running results.
- **Deep learned depth map:**
  Monocular visual odometry approaches are prone to scale drift and require sufficient motion parallax in successive frames for motion estimation and geometry reconstruction without taking the advantages of depth maps in methods based on RGB-D sensors. Godard et al. [Godard et al., 2017] generate disparity images from RGB-D data, and refine their results on the generated disparity image leading to state-of-art the depth estimation results. Yang [Yang et al., 2018] leverage the estimated depth prediction result in the state-of-art direct monocular visual odometry system DSO and achieve comparable performance to the state-of-art stereo method.
- **Deep learning for loop closure (visual localization):**
  Visual-based localization (VBL) consists of retrieving the pose (position and orientation) of a visual query material within a known space representation. For instance, recovering the pose of a camera that took a given photography according to a set of geo-localized images or a 3D model is a simple illustration of such a localization system [Piasco et al., 2018]. In SLAM system, visual localization serves as a loop closure detector when the tracking failed or as the anchor to perform global optimization across all the estimated poses from the last same place. The classical method (e.g., ORB-SLAM [Mur-Artal and Tardos, 2017]) extract the ORB features when reading in every image and store the ORB features in (Visual) Bag of Words (BoW) [Gálvez-López and Tardós, 2012]. While performing place recognition, the BoWs of target images are matched with 3D points in the estimated map to determine if the camera has returned to the same place. In 2015, Kendall et al. [Kendall et al., 2015] proposed to learn an end-to-end neural network to predict the pose of an input

image. Learned from the generated data from structure from motion (SFM), the network regresses the 6D pose of the input image. To account for the challenging illumination changes or scenery changes in different daytime or weathers, Porav [Porav et al., 2018] use a cyclic GAN to perform the domains transformation in the first place, and classical feature matching is performed in the transformed image domain. Unlike them, we propose to learn high-dimensional feature maps without transforming the original images.

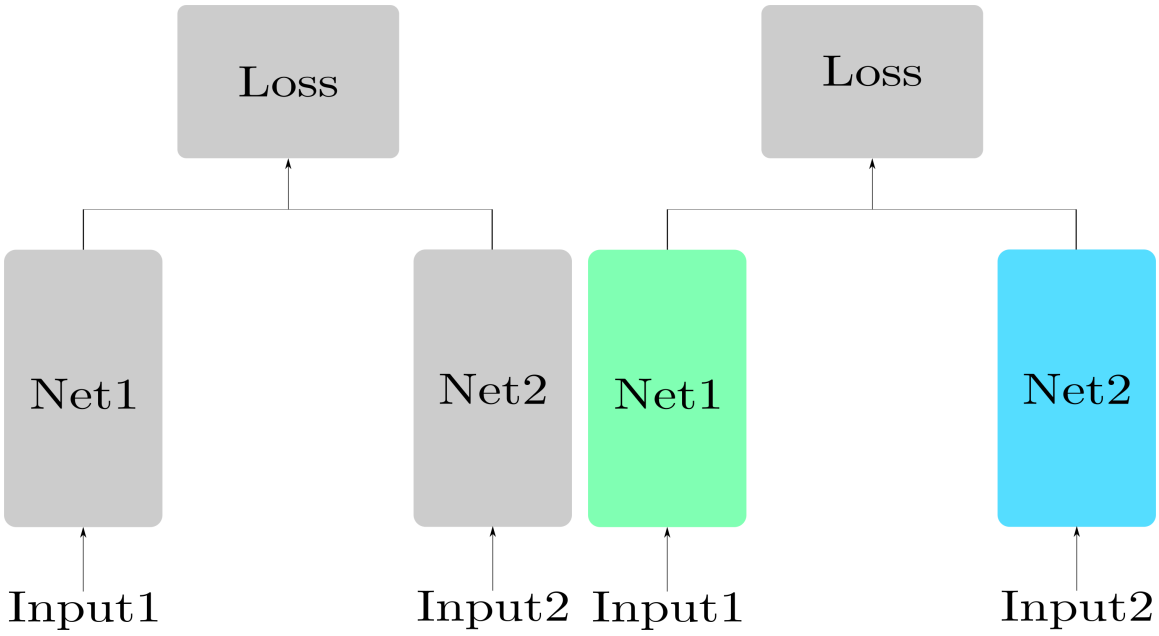
2.2 Deep Learning for Correspondence Learning

Deep dense correspondences: The feature point descriptor learning works based on image patches have shown that it is possible to learn compact image descriptors that significantly outperform handcrafted methods such as SIFT or SURF [Luo et al., 2019, Balntas et al., 2016, Hoffer and Ailon, 2015, Zagoruyko and Komodakis, 2015]. However, the above methods have different shortcomings like the difficulty of defining a universal good margin M in the contrastive loss and the vanishing gradient due to the easy negative pairs in the triplet loss. To address the above mentioned problems, Choy et al. [Choy et al., 2016] published a method to learn a dense map of correspondences from the input images. In their work, a novel correspondence contrastive loss is adopted and thousands of positive correspondences and ten times more of negative correspondences can be used for each batch, such that the CNN converges much faster for a single image pair, allowing faster test time. Similarly, Wohlhart et al. [Wohlhart and Lepetit, 2015] introduce a method that uses a CNN to map the input image to a compact and discriminative descriptor. It uses a novel triplet loss $L = \max\left(0, 1 - \frac{\|s_i - s_k\|_2}{\|s_i - s_j\|_2 + m}\right)$ to enlarge the Euclidean distance between two different objects and between the same objects from different views. The formulation of the triplet loss does not suffer from vanishing gradients when the distance of dissimilar pairs is very small, learning a Mahalanobis distance [Hoffer and Ailon, 2015, Wang et al., 2014, Mishchuk et al., 2017, Piasco et al., 2019]. Inspired by [Choy et al., 2016], Schmidt et al. [Schmidt et al., 2016] train a fully convolutional network (FCN) with a contrastive loss to produce viewpoint- and lighting-invariant dense descriptors, achieving superior single-frame localization results without using any labels to identify correspondences between separate videos. In [Florence et al., 2018] the authors train a deep neural network using the similar pixel-wise contrastive loss but aiming to find consistent dense correspondences across objects with data association provided by 3D geometry. The dense map of correspondences is object dependent and it is used for robotics manipulation. Based on the pixel-wise contrastive loss and aiming to improve robustness for larger baselines, Stumberg et al. [von Stumberg et al., 2020] propose the Gauss-Newton loss function, which has a probabilistic derivation, and the authors apply their resulting features to pose estimation. In this way, they outperform the state-of-art direct and indirect methods for SLAM on their proposed task of relocalization tracking.

Chapter 3

Background

We are interested in learning a dense visual descriptor mapping which is a non-linear function mapping a full-resolution RGB image to a dense descriptor, namely from $R^{W \times H \times 3}$ to $R^{W \times H \times D}$, i.e., mapping each pixel to a vector of length D. To preserve similarity a siamese architecture [Bromley et al., 1994] is used for training. In this chapter we give a short introduction about the siamese network and our selected network, the **U-Net** [Ronneberger et al., 2015], which will serve as the branch for the **siamese network**.



(a) Siamese network (b) Pseudo-siamese network

Figure 3.1: (a) A siamese network has two equal branches sharing their weights, (b) while a pseudo-siamese network has two different branches that do not share weights.

3.1 Siamese Network and Triplet Network

Network Structure: In a siamese network two identical neural networks are placed in parallel, each branch in a siamese network maps the input into new space and presents the information in the new space, and the final outputs in the new space are fed into another neural network and then their similarity is compared either by minimizing an Euclidean distance, or directly using a similarity loss such as contrastive loss, which will be explained in Section 4.1.2, thereby the similarity of two inputs are compared. The input of a siamese network is a pair of positive samples or a pair of negative samples during each training step, depending on the similarity of the two samples. The basic structure of a siamese network is shown in Fig. 3.1(a).

While siamese networks use two equal branches sharing weights to learn the similarity, one variant is to use two different network as branches without sharing the weights, called **pseudo-siamese network**, shown in Fig. 3.1(b). The pseudo-siamese network increases the number of parameters, generally with a large complexity. On the contrary, this also makes it more flexible. Benefiting from its un-similarity in network structure, pseudo-siamese networks are used to learn the differences in images, e.g. in [Mou et al., 2017] a pseudo-siamese network was used to perform change detection in images and in [Hughes et al., 2018] the hard negatives mining is performed via a pseudo-siamese GAN.

If one additional branch is added, the network then becomes a **triplet network**, as shown in Figure 3.2. The input of a triplet network is a triplet consisting of an anchor sample x_a, a positive sample x_+ and a negative sample x_-. These three samples relate to each other by a similarity relationship, i.e., the positive example is more similar to the anchor than to the negative sample.

Objectives: Let $f(x)$ be the mapping function of each branch, for two positive samples x_a^+ and x_b^+ and two negative samples x_a^- and x_b^+, the objective of siamese network is formulated as following:

$$\mathcal{L} = \left(s_i \|x_a^+ - x_b^+\|_2^2 + (1 - s_i) \max\left(0, M - \|x_a^- - x_b^-\|_2\right)^2 \right) \tag{3.1}$$

For positive samples we denote $s_i = 1$ and for negative samples $s_i = 0$, M is the margin. This loss function is called **contrastive loss**, which aims at minimizing the distance between positive correspondences and maximizing the distance between the negative correspondences from at least margin M.

Different from the contrastive loss, the **triplet loss** only constrains the distance of positive correspondences and negative correspondences to be margin M apart. Take a triplet (x_a, x_+, x_-) as input and $f(x)$ be the mapping function of each branch, the triplet loss is formulated as follows:

$$\mathcal{L} = \left(\|x_a - x_+\|_2^2 + \max\left(0, M - \|x_a - x_-\|_2\right)^2 \right) \tag{3.2}$$

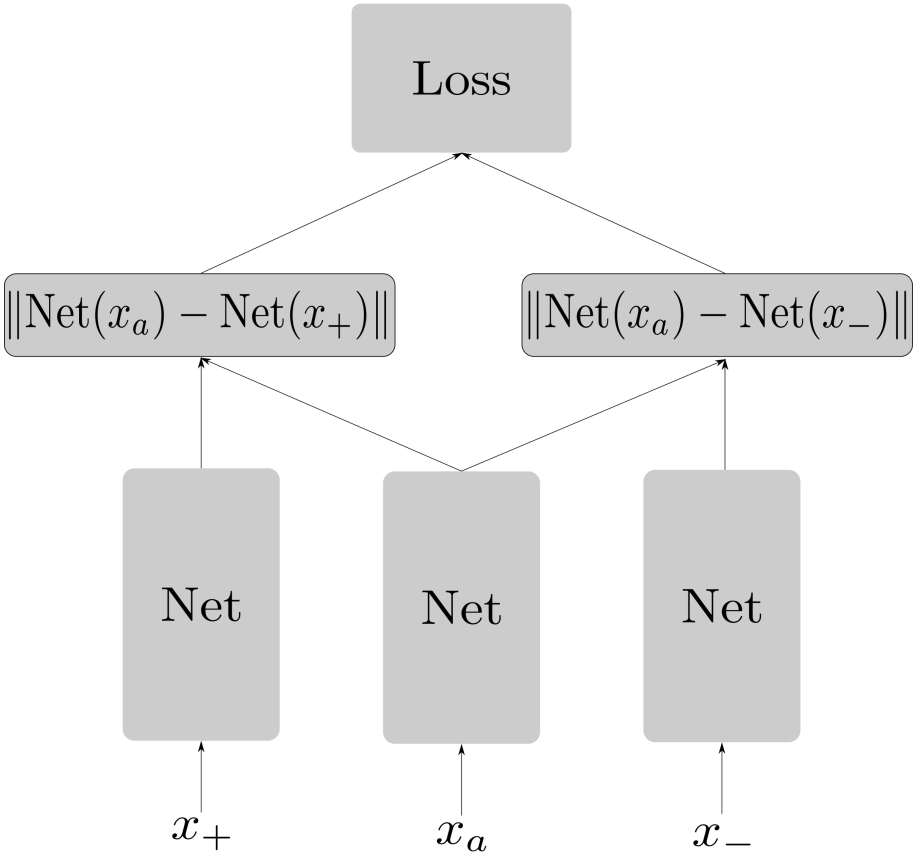


Figure 3.2: The triplet network takes a triplet (x_a, x_+, x_-) as input, while sharing the weights.

Applications: Siamese networks and triplet networks have been widely used to learn similarity in different tasks, which we summarize as **similarity learning**. Similarity learning aims at learning to produce consistent representations on the basis of the similarity or dissimilarity of training data. They are employed in many areas like face verification, person re-identification, and descriptors learning. There is a vast corpus of previous works that use siamese architectures of CNNs to learn the similarity, from which we next select the relevant ones. Sun et al. [Sun et al., 2014] presented a joint identification-verification approach for learning face verification with an Euclidean distance-based contrastive loss. Novel Deep IDentification-verification features (DeepID2) are compared for different faces. FaceNet [Schroff et al., 2015] in turn uses a triplet network that combines the Inception network and an 8-layer CNN which learns to align face patches during training in order to perform face verification, recognition, and clustering. The method trains the network on triplets of increasing difficulty using a negative example mining technique. Another success of triplet network is in person re-identification, Hermans et al. [Her-

mans et al., 2017] proposed to use triplet network and actively mining the hardest positives and hardest negatives mining.

More relevant work in our context is [Simo-Serra et al., 2015], the authors use a siamese network to learn a discriminative representation of patches from different views of a 3D pointcloud; they use the subtractive normalized CNN outputs of the siamese networks as descriptor, and learn a discriminative result with the contrastive loss. Similar work in [Balntas et al., 2016] also uses the feature output of the network to represent the learned descriptor without any metric layers, but using a triplet loss. Similarly, Hoffer et al. [Hoffer and Ailon, 2015] add a metric layer after the feature outputs from a CNN; the metric layer serves as a L2 distance layer between the embedded representation, which is also introduced by Zagotuyko [Zagoruyko and Komodakis, 2015] and is reported to perform better than generic L2 matching, while leveraging different variants of siamese networks.

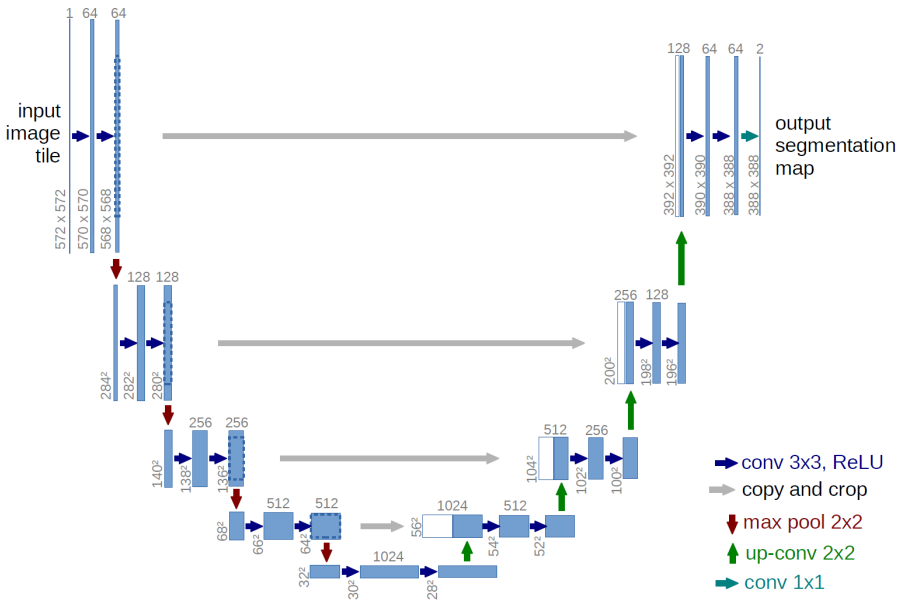


Figure 3.3: The original U-Net architecture; the lowest resolution is 32×32.

3.2 Fully Convolutional Neural Network

The objective of our work is to learn same-sized feature maps as the original input images. To achieve this goal, the network of each branch should have such decoders, whose final output is the same size as the input. We choose the **U-Net** [Ronneberger et al., 2015] for each branch. Next we will give a brief introduction of the U-Net.

The U-Net was firstly used for the segmentation of medical imaging. The original network (as shown in Figure 3.3) consists of a down-sampling path (encoder) and an up-sampling path (decoder). The encoder follows the typical architecture of two 3×3 convolutions, each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for down-sampling. In the original U-Net, the convolution is unpadded, which resulted in the loss of a pixel on each side after every convolutional layer. In contrast to that, we pad the convolution block with zeros to preserve the size of the feature maps to be same for the convolution operation. Meanwhile, inspired by [Ioffe and Szegedy, 2015], we add batch normalization (BN) layers after each 3×3 convolution block to get rid of the influence of internal covariate shift, which accelerated the training process in our experiment. After each down-sampling, the channels of feature maps are doubled. For the up-sampling part, the novel idea is to use a symmetric structure as the encoder part, the only difference is that for the up-sampling operation, the author used transposed convolution operation and then concatenated with the output of the convolutional layer from same level of encoder. At the final layer, a 1×1 convolution is used to map each component feature vector to the desired number of classes.

The U-Net has been widely used in many applications. Many new network designs have been using the core idea of the U-Net: an Encoder-Decoder structure with intermediate concatenation. In oder to see deeply why an U-Net-like structure is so popular, we summarize the reasons in the following:

- The U-Net is a fully convolutional neural network (FCN) with encoder-decoder architecture, which has several benefits.
  - First, since there is no fully-connected layer, this network can map arbitrarily large images to desired dense descriptors.
  - Second, for overlapping regions some of the activations can be reused, which reduces the computational cost.
- The skip connections provide local information from the down-sampling path with the contextual information in the up-sampling path to finally obtain a general information combining localization and context, which is necessary to perform a pixel-wise prediction.
- The structure of the U-Net is symmetric, which provides more convolution layers in the up-sampling path, thus more latent information can be learned.

Chapter 4

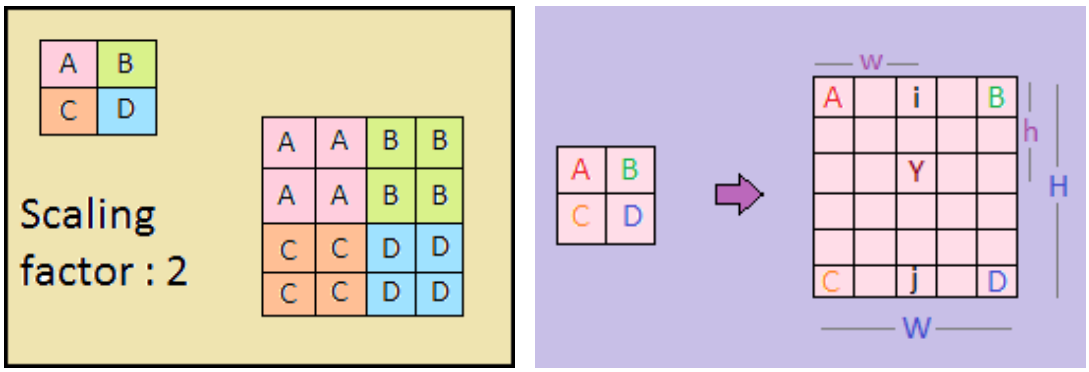
Technical Approach

In this chapter we will explain our approach in detail, this chapter is divided into two parts. In the first part we will compare different techniques used in up-sampling in decoder and present the detailed network structure. Then we will discuss different losses and proposed a new loss formulation to the best use in our task. Finally we will study the pros and cons of different sampling methods and propose our sampling method focusing on our task.

4.1 Proposed Method

4.1.1 Network Architecture

The structure of our network is shown in Figure 4.6.



(a) Nearest Neighbor Interpolation (b) Bilinear Interpolation

Figure 4.1: Different methods of interpolation.

Up-sampling: The encoder part of our own implementation follows the original U-Net structure, which will be described in the Implementation Section 4.2. For the decoder part we use different up-sampling methods. In the original U-Net paper,

Ronneberger [Ronneberger et al., 2015] use the stride-half convolution, also known as deconvolution, to up-sample the features of low resolution to double-sized features. Fig. 4.2 gives an example for a deconvolution with 3×3 kernel and a half stride. From the leftmost figure it can be observed that 8 out of 9 pixels are zeros, but still they contribute to the computation, wasting computational resources. Other than that, if we take a look at the third figure from left to right, it is notable that 2 out of 9 pixels are non-zeros, while in the leftmost only 1 out of 9 is non-zero, which means that different amounts of information enter the computation at different parts of the deconvolution, which is not optimal and yields artifacts.

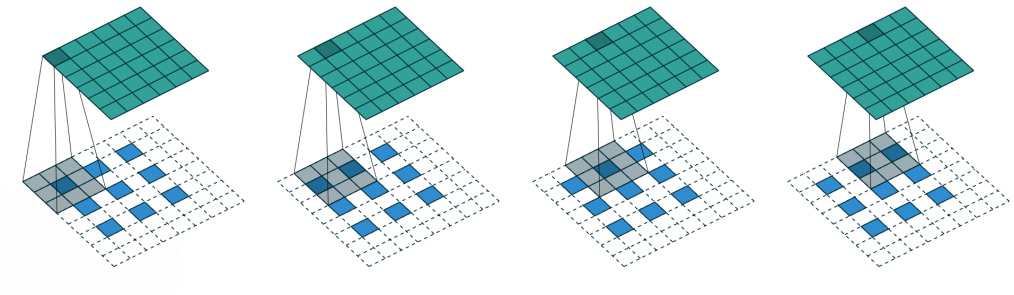


Figure 4.2: An example for 3×3 kernel stride-half convolution, namely double sized up-sampling.

In recent years, it has been widespread to use the Nearest Neighbor Interpolation and the Bilinear Interpolation [von Stumberg et al., 2020], which can be seen in Figure 4.1. In the Bilinear Interpolation method the spaces are filled with the interpolated values of the nearby pixel values. First by interpolating between i and j:

$$i = A + w * \frac{B - A}{W} \tag{4.1}$$

$$j = C + w * \frac{D - C}{W} \tag{4.2}$$

and then by interpolating Y:

$$Y = i + h * \frac{j - i}{H} \tag{4.3}$$

For both interpolations a 1×1 convolution always follows in order for the net to be able to learn the best presentation of the interpolated features maps with high resolution.

Another novel idea called Sub-Pixel Convolution is proposed by Shi et al. in 2016 [Shi et al., 2016], in which they rearrange every pixel in each grid cell into the little $r \times r$ grids that are located throughout, following a normal convolution. The last two

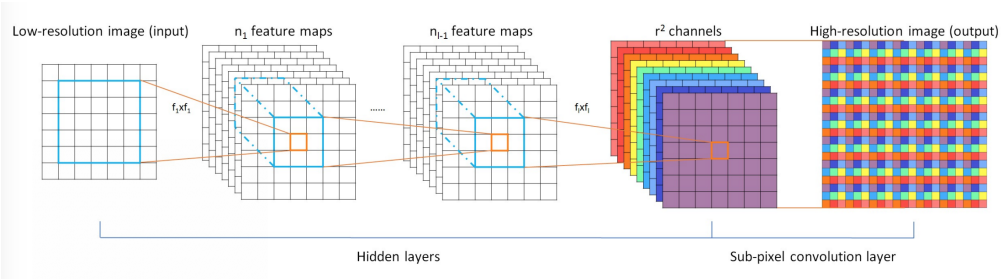


Figure 4.3: Pixel shuffle operation for $r = 3$.

steps in Fig. 4.3 show Sub-Pixel Convolution; the elements of $H \times W \times r^2C$ are rearranged to form $rH \times rW \times C$ feature channels. Before the pixel shuffle operation, a convolution with the size of weights $C \times r^2C \times k \times k$ is performed in the lower resolution space. Compared with the above mentioned interpolation method, this requires less activation calculations since the interpolation is followed by a convolution in high resolution space, and it performs r^2 times more computation of activations. Besides the reduction of the computation, a recent work based on the Attention Residual Network for image classification [Liang et al., 2019] also reports some performance improvement; they report a test error reduction by 0.17% using pixel shuffle up-sampling compared with interpolation methods.

Checkerboard artifacts: A major drawback for deconvolution are the checkerboard artifacts. This happens when the deconvolution has an "uneven overlap". In particular, as shown in Figure 4.4 [Odena et al., 2016], deconvolution has an uneven overlap when the kernel size (the output window size) is not divisible by the stride (the spacing between points on the top). While the network could, in principle, carefully learn weights to avoid this result, in practice neural networks struggle to avoid it completely.

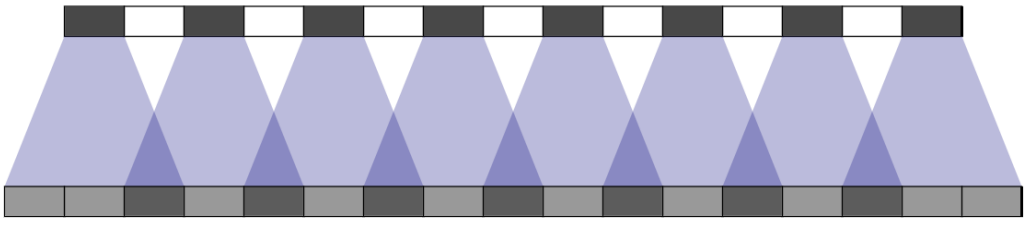


Figure 4.4: Uneven overlap for kernel size 3 and stride 2.

In the up-sampling path of the U-Net, the up-sampling blocks are repeated for multiple times (4 times for our implementation), iteratively building up the feature

maps with the same size as the input image from the bottleneck of the lowest resolution descriptions. These stacked deconvolution operations in practice compound and create artifacts on a variety of scales, which are illustrated in Figure 4.5 [Odena et al., 2016].

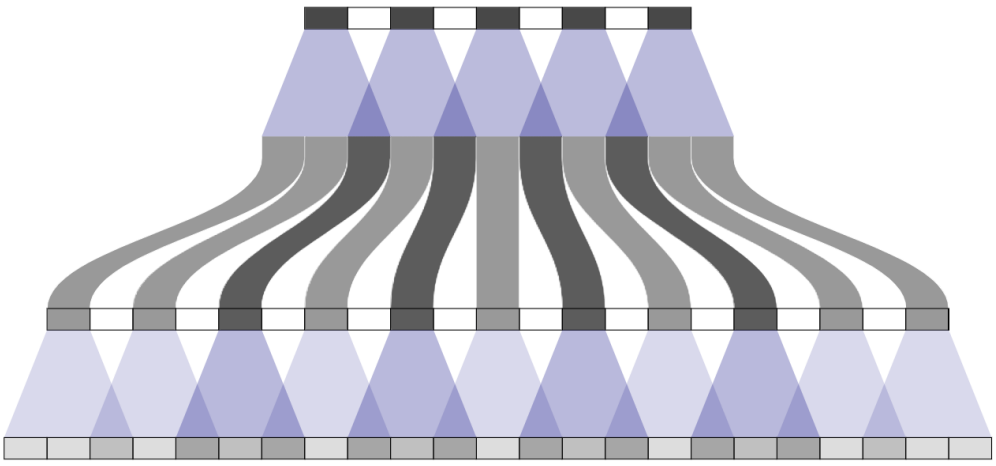


Figure 4.5: Stacked uneven overlaps for kernel size 3 and stride 2.

In the pixel shuffle convolution operation similar checkerboard artifacts are observed, but because of the repeating initialization problem when the convolutional kernel is initialized, the 9 pixels in this 3×3 grid are going to be random values and thus totally different. While the next set of 9 pixels are randomly different to each other, they are very similar to their corresponding pixels in the previous 3×3 section. As the network tries to optimize the weights, it is starting from this repeating 3×3 starting point, leading to repeated 3×3 patterns, as shown in Figure 4.3. To avoid these checkerboard patterns, Aitken et al. [Aitken et al., 2017] propose a novel initialization method which is named ICNR. In order to keep these 3×3 pixels the same, one channel out of the r^2 sets of channels is randomly initialized; then, in the next step these randomly initialized weights are copied to the remaining $r^2 - 1$ channels. In this way, each of the 3×3 regions will be equal.

The results of the three different up-sampling techniques will be demonstrated in Section 5.

4.1.2 Loss Function

As mentioned before, there exist two types of losses to learn dense correspondences: contrastive loss [Choy et al., 2016, Florence et al., 2018, von Stumberg et al., 2020, Schmidt et al., 2016] and triplet loss [Wohlhart and Lepetit, 2015]. Triplet losses are generally reported to outperform siamese losses, sometimes by a large

margin. Hoffer and Ailon [Hoffer and Ailon, 2015] report that siamese losses failed completely in their experiments while triplet losses performed well. To give this an illustration, suppose we are given faces of two different people, one is smiling, while the other one is upset, and we tell to the net that these are negative (dissimilar) samples. The question that arises here is: what aspect makes them dissimilar? Is it because the images show two different people? Or is it because of the different emotions? If we had some additional information telling that a smiling face and a third face with a smiling emotion are similar, then it would be clear that we are more interested in putting the faces with similar emotions together [Keller et al., 2018]. This example indicates that only pairs of samples will fail in some tasks due to the lack of context. Triplets add context by removing ambiguity.

Next we will formalize of two losses from our perspective of view and discuss the advantages and disadvantages of both losses.

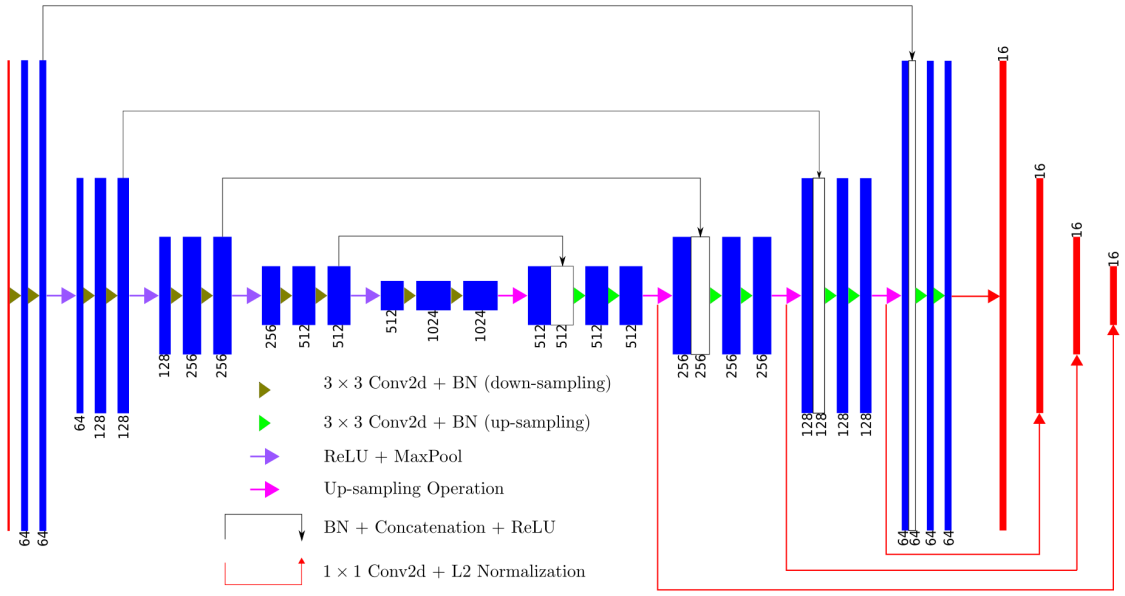


Figure 4.6: Detailed structure of our network. Note that we take out every level of feature maps by a 1×1 convolution operation to leverage the hierarchical feature maps of U-Net.

Contrastive loss and triplet loss: Given 2 input images I_a and I_b, the two U-Net branches will output the pyramid feature maps of F_a^l and F_b^l, in which l is the different scale level in the output feature pyramid. The contrastive loss takes the features at coordinates x_i^a and x_i^b, $F_a^l(x_i^a)$ $F_b^l(x_i^b)$. If x_i^a and x_i^b correspond to the same 3D vertex, we treat this pair as a positive pair and encourage them to be close in feature space; otherwise we want them to be at least margin M apart [Choy et al., 2016]. The full loss is calculated as follows:

$$\mathcal{L}_{\text{contr}} = \sum_{l} \left(\frac{1}{N_{\text{pos}}} \sum_{i}^{N_{\text{pos}}} s_i {D^l_{\text{feat}}}^2 + \frac{1}{N_{\text{neg}}} \sum_{i}^{N_{\text{neg}}} (1 - s_i) \max\left(0, M - D^l_{\text{feat}}\right)^2 \right) \tag{4.4}$$

where $D^l_{\text{feat}}(\cdot)$ is the L2 Euclidean distance between the features at two coordinates: $D^l_{\text{feat}} =\| F^l_a(x^a_i) - F^l_b(x^b_i) \|_2$; for positive pairs we denote $s_i = 1$ and for negative pairs $s_i = 0$.

The triplet loss takes a triplet as an input. A triplet (t_a, t_p, t_n) is formed such that the following condition is fulfilled: t_a, t_p stem from the same 3D vertex, and t_a, t_n from the different one. The triplet loss can be defined as follows:

$$\mathcal{L}_{\text{triplet}} = \sum_{l} \left(\frac{1}{N_{\text{triplets}}} \sum_{i}^{N_{\text{triplets}}} \max\left(0, D^l_{\text{ap}} - D^l_{\text{an}} + M\right)^2 \right) \tag{4.5}$$

in which $D^l_{\text{ap}}(\cdot) =\| t_a - t_p \|_2$ and $D^l_{\text{an}}(\cdot) =\| t_a - t_n \|_2$ present the distances between the positive pairs and negative pairs.

Discussion: One potential drawback of the contrastive loss is that a constant margin M has to be selected for all pairs of negative samples. This leads to the problem that all the negative pairs corresponding to the different 3D points are embedded in the same small space as the visually similar ones, despite different viewpoint, lighting condition, weathers etc. [Wu et al., 2017]. In contrast, the triplet loss merely tries to keep all positives closer to any negatives for each triplet, and does not impose a constant margin M, which allows the embedding space to be arbitrarily distorted.

However, in triplet loss another problem arises. In the process of learning dense correspondence maps, we would like to have the positive pairs as close as possible; while contrastive loss will limit the distance between positives as close to zero as possible, triplet loss only considers to push the distances between positives and negatives as far as possible, until a margin M.

Chen et al. [Chen et al., 2017] proposed an improved loss function based on the triplet loss, coined quadruplet loss, which aims to push away negative pairs from positive pairs with respect to different pairs:

$$\begin{aligned} \mathcal{L}_{\text{quad}} = \sum_{l} \left(\frac{1}{N_{\text{triplets}}} \sum_{i}^{N_{\text{triplets}}} \max\left(0, D^l_{\text{ap}} - D^l_{\text{an}} + M_1\right)^2 \right) + \\ \sum_{l} \left(\frac{1}{N_{\text{triplets}}} \sum_{i}^{N_{\text{triplets}}} \max\left(0, D^l_{\text{ap}} - D^l_{\text{nn}} + M_2\right)^2 \right) \end{aligned} \tag{4.6}$$

The first term is the same as in Eq. (4.5), which focuses on the relative distances between positive and negative pairs w.r.t. the same feature points; the second term is the new constraint considering the maximization of the distance between negative-negative pairs and anchor positive pairs to a particular margin; and by negative-negative pairs we mean the negatives sampled on each image.

The question will arise of whether the Euclidean norm is the right metric of the real difference of the samples. As well known, to compare two high-dimensional vectors, we have these different criteria:

$$\mathcal{D} = \|x_a - x_b\|_2 \tag{4.7}$$

$$\mathcal{D} = \|x_a - x_b\| \tag{4.8}$$

$$\mathcal{D} = \frac{{x_a}^T x_b}{\sqrt{{x_a}^T x_a {x_b}^T x_b}} \tag{4.9}$$

The first two forms are widely used in deep metric learning. The form in Eq. (4.8) is the absolute distance, which has been used in [Hao et al., 2018], but the shortcoming of the absolute loss is that it is not differentiable at some point, more specifically when the distance is equal to zero. In Eq. (4.7) (Euclidean norm) the problem is that it cannot distinguish the magnitude of the vectors, but this is not a problem in our case, since we have L2 normalized final feature vectors. The final distance could be simplified to

$$\mathcal{D} = {x_a}^T x_b \tag{4.10}$$

However in the experiment, we find that for these two different criteria, the final performance dose not make much difference.

Our objectives: Different from metric learning or patch matching in the similarity learning task, where in a triplet (t_a, t_p, t_n) each element is unique, e.g. in the task of face recognition, t_a is the face of one person, t_p is the face of the same person but from a different view of perspective, and t_n is the face of a different person, or in patch matching task, t_a could be a patch which include a pattern of paintings, and t_p is the same pattern but in a different view of scale or different lighting, t_n is the patch containing a different pattern. In contrast, in our task, since we are matching the correspondence of 2 different images. Taking 2 images I_a, I_b, and the positive pairs x_+^a and x_+^b and x_-^a and x_-^b are on the corresponding images, in this case, the positive part(t_a, t_p) in triplet are x_+^a and x_+^b, but for the t_n there are two directions:

- If we treat I_a as the anchor image, then the anchor is x_+^a, positive triplet is x_+^b, the negative triplet is then x_-^b. $D_{\text{ap}}^l(\cdot) = \| x_+^a - x_+^b \|_2$, $D_{\text{an}}^l(\cdot) = \| x_+^a - x_-^b \|_2$, and we define $\mathcal{L}_{a\to b}$ as triplet loss from I_a to I_b:

$$\mathcal{L}_{a\to b} = \sum_l \left(\frac{1}{N_{\text{triplets}}} \sum_i^{N_{\text{triplets}}} \max\left(0, \| x_+^a - x_+^b \|_2 - \| x_+^a - x_-^b \|_2 + M\right)^2 \right) \tag{4.11}$$

- If we treat I_b as the anchor image, then the anchor is x_+^b, positive triplet is x_+^a, the negative triplet is then x_-^a. Similar to (4.11) we define triplet loss from I_b to I_a : $\mathcal{L}_{b\to a}$ as:

$$\mathcal{L}_{b\to a} = \sum_l \left(\frac{1}{N_{\text{triplets}}} \sum_i^{N_{\text{triplets}}} \max\left(0, \parallel x_+^a - x_+^b \parallel_2 - \parallel x_+^b - x_-^a \parallel_2 + M\right)^2 \right) \tag{4.12}$$

We formulate our objective for triplet loss as

$$\mathcal{L}_{\text{ab}} = \frac{1}{2}(\mathcal{L}_{a\to b} + \mathcal{L}_{b\to a}) \tag{4.13}$$

Now consider the positive part of contrastive loss, the objective is to pull the positive pairs as close as possible, which leads to the problem of generalization: If two positive pairs are pulled together, the pixels in its neighbor are also pulled together, resulting in the same feature vectors value near the positive pairs, thus affecting the re-projection errors in direct SLAM. Instead of pulling the positive pairs as close as possible, we aim to learn to pull the positives as close in a margin as possible, which are formulated as following:

$$\mathcal{L}_{\text{pos}} = \sum_l \left(\frac{1}{N_{\text{pos}}} \sum_i^{N_{\text{pos}}} (D_{\text{feat}}^l - M_{\text{pos}})^2 \right) \tag{4.14}$$

The explanation of two different positive objectives is shown in Fig. 4.7.

The full objective is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{\text{pos}} + \lambda \mathcal{L}_{ab} \tag{4.15}$$

4.1.3 Hard Negative Mining

From Eq. (4.4) we notice that not all negative pairs contribute to the final hinge loss. Only those pairs, for which the Euclidean norm of the difference between the negatives is smaller than the margin M, take some effect on the back-propagation of the negative loss. The same problem appears in Eq. (4.5). According to the different distance boundaries the triplet can be divided into three groups:

1. **Easy triplets** where $D_{\text{ap}} + M < D_{\text{an}}$; it will contribute nothing to the triplet loss.

2. **Hard triplets** where $D_{\text{an}} < D_{\text{ap}}$; it is hard to push negative pairs apart from positive pairs.

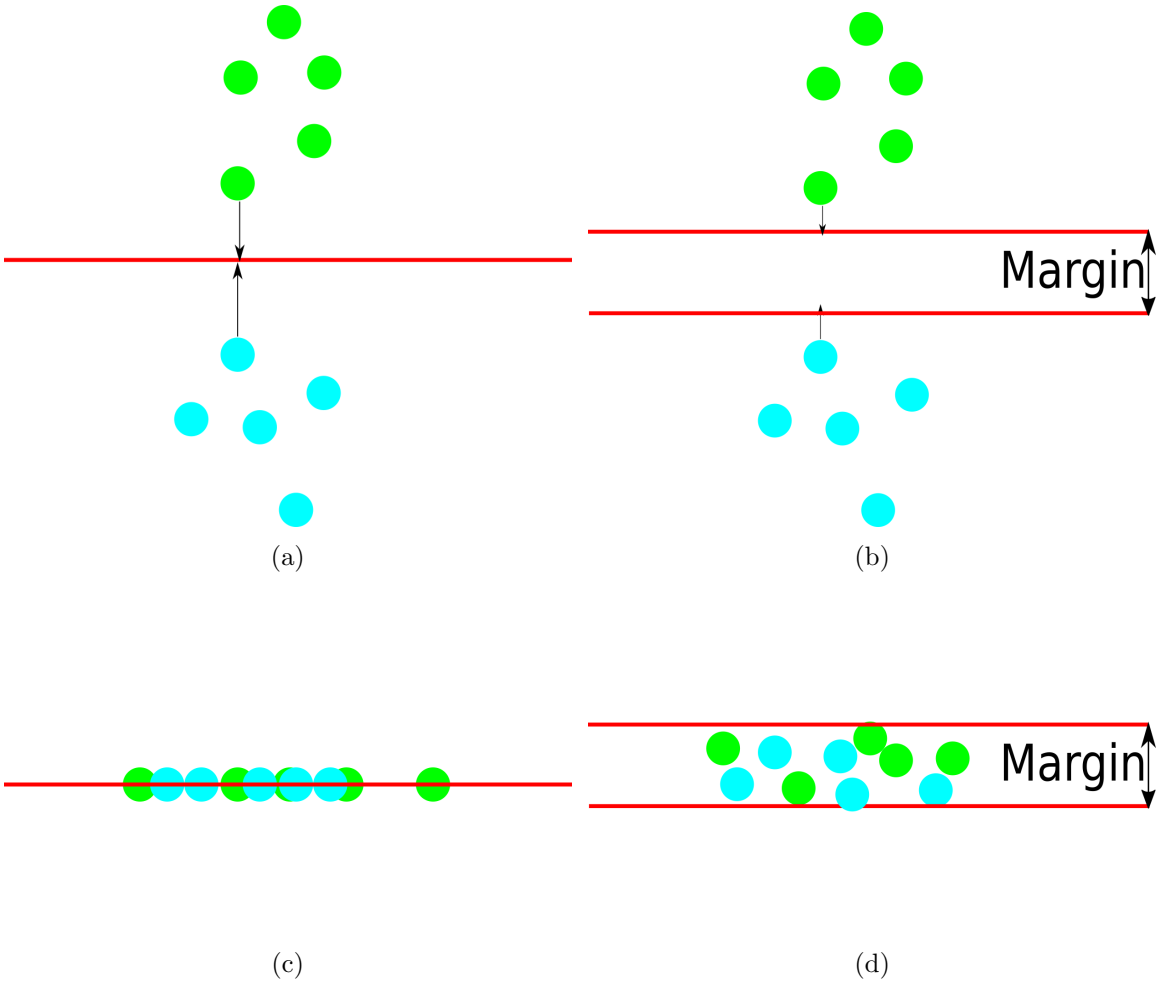
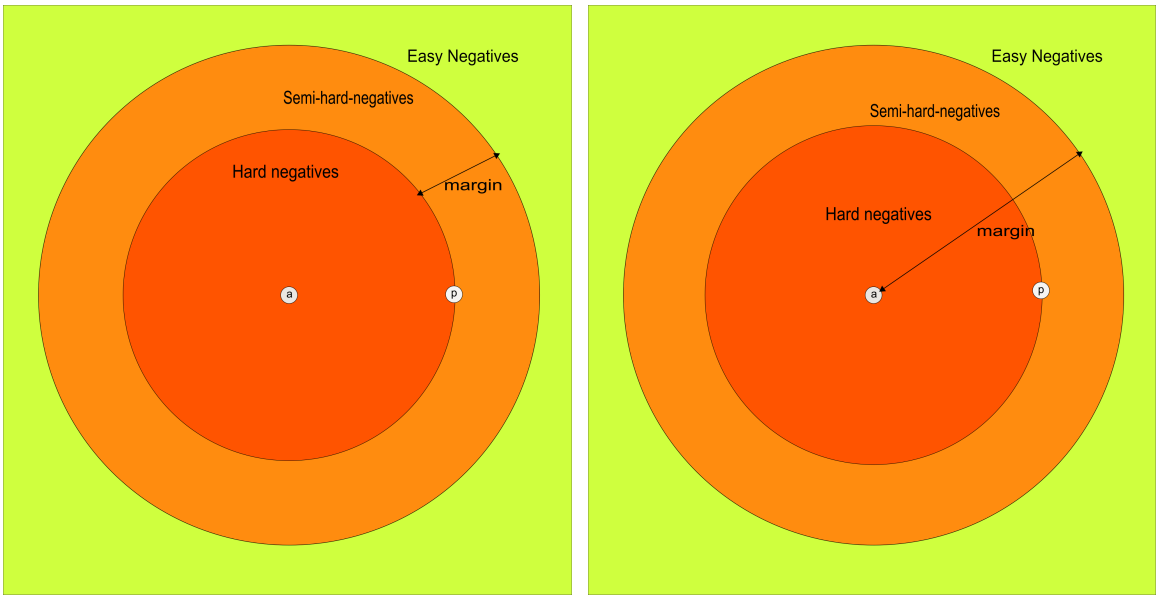


Figure 4.7: Examples for two different positive losses. Left column: (a) shows the original positive loss before learning, (c) shows how positive samples change after learning. Right column: (b) shows the margin-based positive loss before learning, (c) shows how positive samples change after learning. This shows that after the loss has converged, the positives samples are not in the same values.

3. **Semi-hard triplets** where $D_{\text{ap}} < D_{\text{an}} < D_{\text{ap}} + M$; these contribute to the triplet loss and are easier to separate.

An important decision for training with triplet loss is the **negatives selection** or **triplet mining**. The strategy chosen will have a high impact on the training efficiency and final performance. An obvious appreciation is that training with easy triplets should be avoided, since their resulting loss will be zero. Very similarly for contrastive loss, the samples of positive correspondences and negative correspondences can also be divided into three groups, but with a different criteria:

1. **Easy negatives** where $D_{\text{neg}} > M$; negative loss will be zero, thus these negative pairs contribute nothing to the gradient.



(a) Triplet loss (b) Contrastive loss

Figure 4.8: Metric space for triplet loss and contrastive loss.

2. **Hard negatives** where $D_{\text{neg}} < D_{\text{pos}} < M$; these negatives are important; we will benefit most by putting these negatives apart.

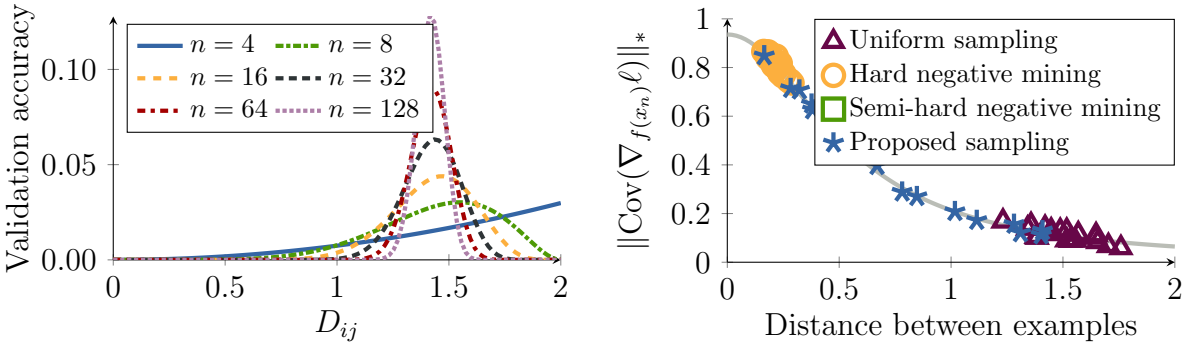
3. **Semi-hard negatives** where $D_{\text{pos}} < D_{\text{neg}}$; these contribute to the negative loss and are easier to separate.

Among all different losses discussed in the last subsection, the different samples can be divided into the above-mentioned classes, as shown in Fig. 4.8. Most works in the literature consider negative mining as a very important step: Wu et al. [Wu et al., 2017] claim that in deep metric learning the sample selection plays an equal or more important role than the loss, and they propose a sampling method leading to a low variance of gradients and thus a more stable training, resulting in an improvement irrespective of the loss function. In [Florence et al., 2018] Florence proposed to uniformly mine only the hard negative samples for cross objects contrastive loss, which are the hard negatives and semi-hard negatives in Fig. 4.8(b). These negatives fulfill the condition $D_{\text{neg}} < M$, making the negative loss always greater than zero:

$$M - D_{\text{neg}} > 0 \tag{4.16}$$

thus the mined negatives will always contribute to the negative loss and accelerate the convergence of the negatives loss.

This mining method is, however, not optimal: once the network has reached a reasonable level of performance, most of the negatives pairs will be discarded, leading



(a) Density of datapoints on the D-dimensional unit sphere

(b) Sample distribution for different strategies

Figure 4.9: (a) The concentration of measurements as the dimensionality increases — most points are almost equidistant. (b) High variance means that the gradient is close to random, while low variance implies a deterministic gradient estimate. Lower is better. It shows the empirical distribution of samples drawn for different strategies. All three approaches are biased towards certain distances; in contrast, our proposed weighted sampling selects a wide range of samples instead.

to a slow convergence near the optimum and effectively stall the learning process. If the hardest examples are selected and are optimized (e.g., pushed away from each other), then the learned hard examples are finally performing well. That can be seen in the result of [Simo-Serra et al., 2015], which shows that final performance is benefited from choosing only the hard negatives.

One crucial issue of selecting the hardest negatives can, however, lead to a collapsed model. Consider a negative pair $t := (x_i, x_-)$ or a triplet $t := (t_a, t_p, t_n)$. The gradient with respect to the negative example $f(x_n)$ is in the form of

$$\partial_{f(x_n)}\mathcal{L}(\cdot) = \frac{1}{D_{\text{neg}}}\omega(t) \tag{4.17}$$

in which $\omega(t)$ is some function, $D_{\text{neg}} := \|f(x_i) - f(x_n)\|_2$ and note that $\frac{1}{D_{\text{neg}}}$ determines the gradient. A problem arises when D_{neg} is small, and the estimates are noisy; introduced enough noise z by the training algorithm, the gradient $\frac{1+z}{D_{\text{neg}}+z}$ is dominated by noise. In FaceNet [Schroff et al., 2015], the authors proposed to select negative samples such that

$$D_{\text{ap}} < D_{\text{an}} < D_{\text{ap}} + M \tag{4.18}$$

that is, they sample the semi-hard triplets.

To fully understand what good sampling depends on, we should firstly take a dive into the distribution of samples. Recall that the sampled features $F(x_i)$ are constrained to the D-dimensional unit sphere S^{D-1} for large $D \geq 16$. Consider the

situation where the points are uniformly distributed on the sphere. Then the distribution of pairwise distribution follows [Lehnen and Wesenberg,]

$$q(d) \propto d^{n-2}\left(1-\tfrac{1}{4}d^2\right)^{\frac{n-3}{2}} \tag{4.19}$$

Fig. 4.9(a) [Wu et al., 2017] shows a concentration of measures occurring. In fact, in high dimensional space, $q(d)$ approaches $N\left(\sqrt{2}, \frac{1}{2n}\right)$. In other words, if negative examples are scattered uniformly, and we sample them randomly, we are likely to obtain examples that are away by $\sqrt{2}$. Figure 4.9(b) compares the above mentioned sampling strategies. We can see that the above mentioned uniform sampling [Florence et al., 2018] yields only easy examples and induces near no loss, thus stalling the learning process; due to the narrow distance between anchor and negatives, in hard negative mining the gradient is dominated by noise, which is why the samplings lay mostly on high variance areas. Semi-hard negative mining finds a narrow set in-between; after some point, no examples are left within the band, and the network will stop making progress [Schroff et al., 2015].

Proposed Sampling: To avoid the large variance in hard negatives mining and the narrow band in semi-hard negatives mining, we thus proposed a sampling strategy based on the distribution of high dimensional feature vector with respect to distance: We sample uniformly according to distance D_{an}, i.e., sampling with weights $q(d)^{-1}$. In oder to cut off the easy negatives, we set the threshhold to make sure all distances are not so easy to separate, meanwhile, we also clip the weighted sampling to avoid noise samples. To formulate our sampling strategy, we formulate as follows:

$$Pr(n^\star = n|a) \propto max(min(\lambda_{min}, q(d)^{-1}(D_{\text{an}})), \lambda_{max}) \tag{4.20}$$

in which $(a, n^\star)$ is the negative pair, λ_{min} is the minimal threshhold to clip the samples, and λ_{max} is the maximum threshold that removes the easy negatives. The simulated example comparing our proposed method with the above-mentioned three methods is shown in 4.9(b).

4.2 Implementation

4.2.1 Training Data Generalization

In order to generate the training labels under different weather conditions, we formalized our training regime as follows. Assume we have a collection of $\mathcal{V}$ videos under $\mathcal{W}$ different weathers or lighting conditions under different perspectives $\mathcal{S}$, where each $v_i = \{(I_0, D_0, T_0), \cdots, (I_j, D_j, T_j)\}$, where for each frame j we have an image I_j and a corresponding depth map D_j to back-project the image pixels in I_j to vertex map M_j in world coordinates. Specifically, for (u_j, v_i) on I_j, the depth d_i in D_j, and the corresponding P_i in M_j is given as follows:

$$P_i = \pi^{-1}(x_i, d_i) = d_i \left(\frac{u_i + c_x}{f_x}, \frac{v_i + c_y}{f_y}, 1 \right)^T \tag{4.21}$$

In which $\pi(\cdot)$ is the function projecting 3D points onto the image plane and c_x, c_y and f_x, f_y denote the optical center and focal length of the pinhole camera model.

While we have many sequences of the same scenes from different perspectives of views under different weathers, the coordinate system defined by $\mathcal{M}_j \in \mathcal{V}_k$ are not consistent from a different point of view $\mathcal{S}$ but it is consistent for different lighting $\mathcal{W}$. For example, given a video $\mathcal{V}_k$, from different points of view, a leave of a tree on the roadside can map to a particular model vertex under different weather for the CARLA Benchmark dataset and different lighting conditions for the ICL_NUIM dataset [Handa et al., 2014] $\mathcal{W}_a$ and $\mathcal{W}_b$, but in a different perspective of the same tree they will be assigned to a different coordinate depending on their relative transformation T_{initial}.

However, since we have all the relative transformations between the different position of the cameras, we can therefore learn the canonical keypoints. In our case, we always have an image pair and we indicate the positive pairs in pair-wise based loss or anchor-positive pairs in triplet based loss, as those pixels which are projections of the same model point from different videos $\mathcal{V}$.

In practice, we compute dense descriptors for an image with size 512×512. In order to learn the gradient consistent feature maps, we at first extract the Canny edges from the original input images, or high gradient points as DSO [Engel et al., 2016] suggests, and then these candidates are back-projected onto the 3D space, depending on which image plane these 3D model points are projected by applying the transformation T_j

$$P_i^{j_b} = T_{j_b j_a} P_i^{j_a} \tag{4.22}$$

The 3D model points are transformed from the j_a to the j_b frame, and then projected

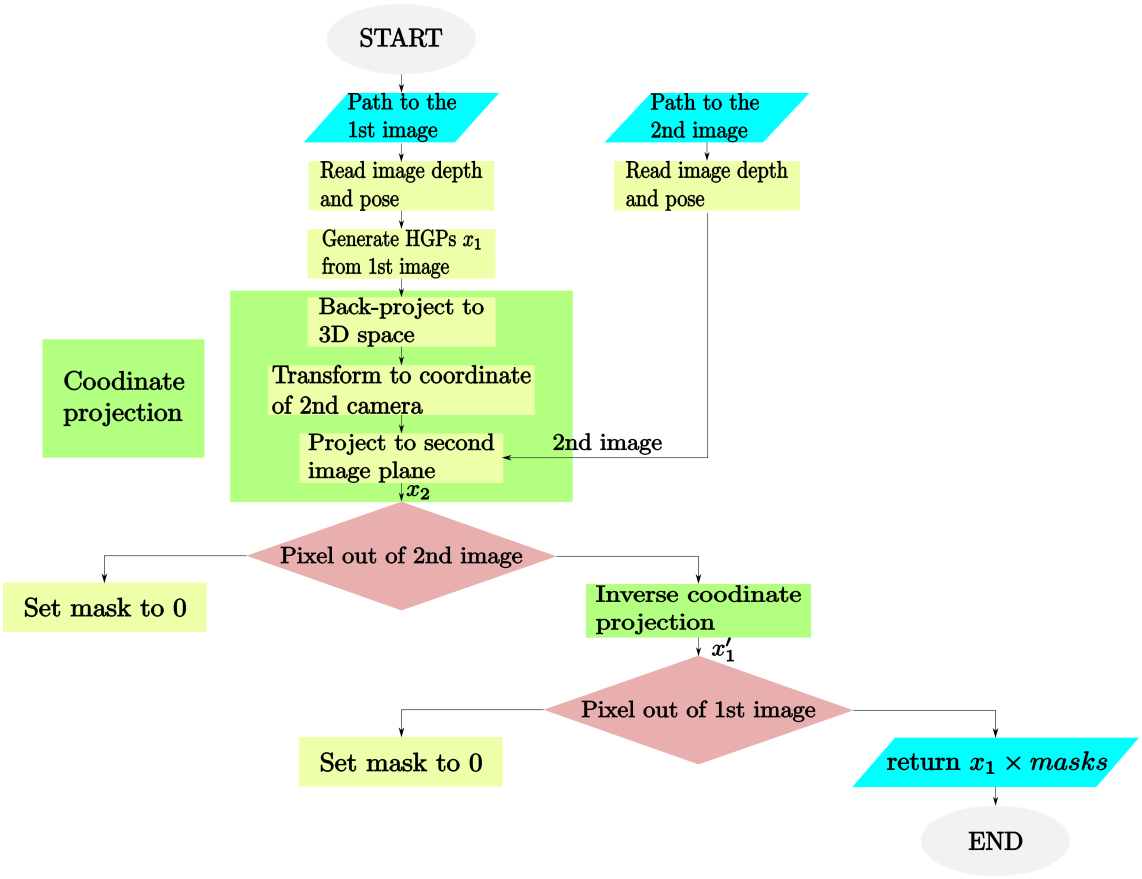


Figure 4.10: Algorithm flow chart to generate training labels.

onto the j_b image frame. Depending on the combination of different upper and lower thresholds, the Canny edge detector results in over 20,000 edge candidates for one single image, for high gradient points, since we select all points whose normalized gradient is larger than 10, and result in about 200,000 candidates, both leading to high computation cost and high demand for GPU direct storage. To save the storage of GPU, we first project all candidates from the first image onto second image, then remove the candidates, which lay out of boundaries (horizontal and vertical), and then do the inverse projection for the left candidates from the second image onto the first image, and remove the candidates that are out of the first image in the same way. This process are illustrated in Figure 4.10. After this, instead of considering all possible candidates, we sample locations at which we can apply different pair-wise losses and triplet losses. We first choose a part of pixel locations x_a (up to 3,000 - 5,000) randomly from the left candidates that have a correspondence in I_{j_b}. For each sample from I_{j_a} we apply the loss function to the pair formed by x_a and the corresponding pixel from I_{j_b}, and then to a certain ratio pair formed by x_a and non-corresponding pixel locations chosen according to the sampling methods as described in Sec. 4.1.3. An example of choosing the training samples is shown in Figure 4.11.

Figure 4.11: Example of the generated samples.

4.2.2 Network Structure

For the encoder part of the U-Net we use a similar structure as the original. As shown in Fig. 4.12, the encoder is composed of 4 levels of different scale feature maps. The down-sampling is a max-pooling with stride 2, after each down-sampling layer

follow 2 convolution blocks, both composed of one 3×3 kernel convolution layer, one batch normalization layer, and one ReLU non-linear activation. For the first convolution block after the max-pooling layer, the channels are doubled to maintain the volume feature numbers from $C_i \times H \times W$ to $C_{i+1} \times \frac{H}{2} \times \frac{W}{2}$ such that $C_{i+1} = 2C_i$. After 4 times down-sampling, the size of the feature maps with lowest resolution is $\frac{1}{16}$ of the original size.

For the decoder part we make some modifications compared to the original version as follows. From the bottleneck, which is the coarsest level, the feature maps are up-sampled with pixel shuffle by 2, and at the same time the number of channels is reduced by 2. The up-sampled feature maps are then concatenated with the feature maps with the same size from the encoder part. After this, we apply D channels 1×1 convolution kernels to get the output filter maps of this level. This is done in an iterative fashion until the finest level, and results in the final output pyramid maps with the sizes $[\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1]$ of the original image size. Then, they are used for deep direct SLAM.

Multi-Scales Loss: To leverage the properties of multi-scale output pyramid maps, we calculate the loss in a pyramidal fashion, namely, we first calculate the positive loss in coarsest level, which is $\frac{1}{8}$ of the finest level, and we iteratively calculate the losses until the finest level, then we sum the average of the loss of each level. For the negative loss we follow the same way. Finally we get the total contrastive loss as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pos}} + \lambda \mathcal{L}_{\text{neg}} \tag{4.23}$$

L2 Normalization and Sigmoid Outputs: We do not give any constraint on the output of the two U-Net branches, which may lead following problems:

- One possible problem is that the output pyramid feature maps of different domain have different ranges, which means the evaluation on these feature maps across different domains are not comparable.

- More problematic is that during the training process, the positive loss and the negative loss are always not in the same range, which results in a negative influence for the gradients.

To alleviate these problems, we tried two different methods: one strategy is that we L2-normalize all the output features from the U-Net branches before calculate the loss, another is simply using a sigmoid or hyperbolic tangent activation at the output layers. Both strategies could solve the first problem, however, when looking in to the possible maximum Euclidean distance between two feature maps $d = \sqrt{\|\boldsymbol{f}(x_1) - \boldsymbol{f}(x_2)\|_2^2}$, we call the distance d_{L2} and d_{sig} for the L2 normalization

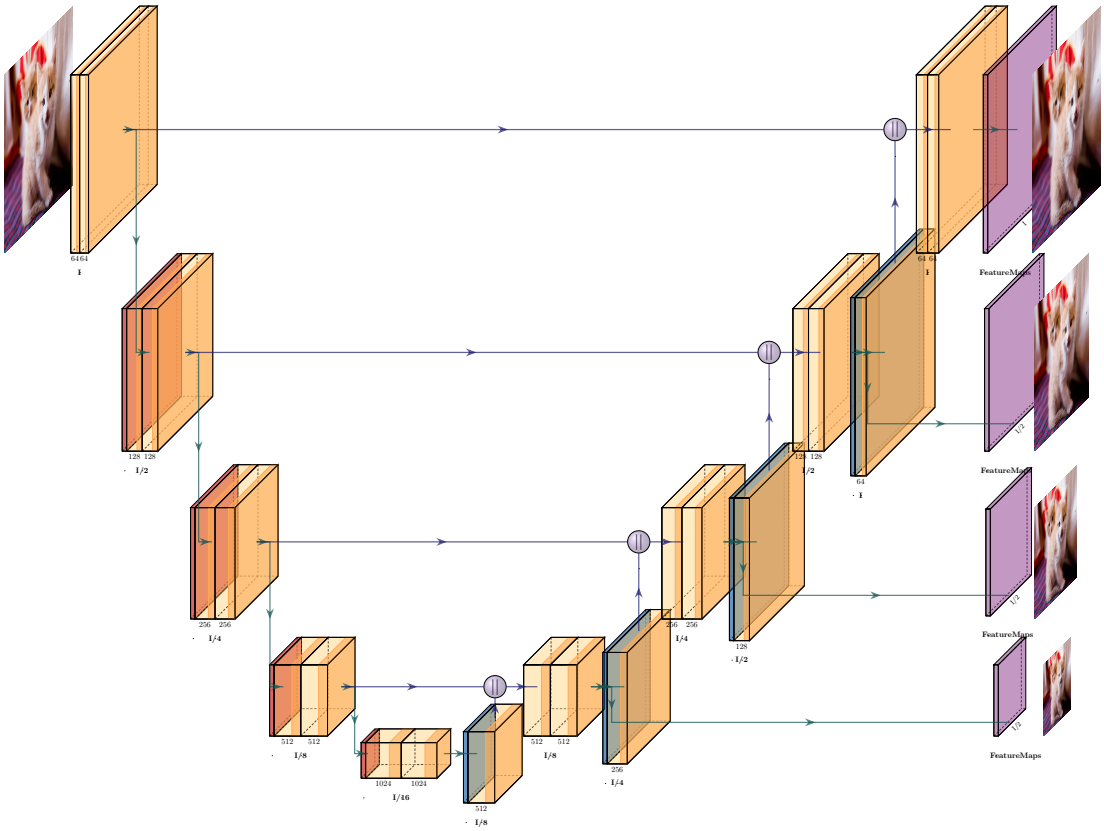


Figure 4.12: The modified U-Net uses the multi-scale properties of the original U-Net decoder now in order to output multi-scale feature maps directly.

method, and for the sigmoid activation method for example

$$\begin{aligned} d &= \sqrt{\|\boldsymbol{f}(x_1) - \boldsymbol{f}(x_2)\|_2^2} \\ &= \sqrt{\boldsymbol{f}(x_1)^T\boldsymbol{f}(x_1) + \boldsymbol{f}(x_2)^T\boldsymbol{f}(x_2) - 2\boldsymbol{f}(x_1)^T\boldsymbol{f}(x_2)} \end{aligned} \tag{4.24}$$

for d_{L2}, since $\boldsymbol{f}(x_1)$ and $\boldsymbol{f}(x_2)$ are both normalized, and

$$d_{\text{L2}} = \sqrt{2 - 2\boldsymbol{f}(x_1)^T\boldsymbol{f}(x_2)} \leq 2 \tag{4.25}$$

As for d_{sig}, each element in $\boldsymbol{f}(x_1)$ and $\boldsymbol{f}(x_2)$ is in range $[0, 1]$, we have:

$$d_{\text{sig}} = \sqrt{\boldsymbol{f}(x_1)^T\boldsymbol{f}(x_1) + \boldsymbol{f}(x_2)^T\boldsymbol{f}(x_2) - 2\boldsymbol{f}(x_1)^T\boldsymbol{f}(x_2)} \leq 2\sqrt{D} \tag{4.26}$$

Where D is the number of channels for the feature maps. We could see that the second method still causes the second problem due to the possible big distance of different features, and the greater the number of channels we choose, the worse this problem becomes. Besides, the sigmoid or tangent activation is non-linear, mapping the outputs non-linearly, which is not desired.

4.2.3 Training and Testing

The training is done from scratch with Kaiming weight initialization [He et al., 2015] to stop the variance of the layers activation from explosion for a very deep network. As an optimizer we choose ADAM with a learning rate of 10^{-6} and a weight decay of 10^{-3}. For the correspondence contrastive loss, we set the Margin $M = 1$, and set the maximum distance of the disturbing point to correct point for the Gauss-Newton loss as the setting in supplementary material of [von Stumberg et al., 2020]. Restricted by the GPU memory, for every training step 2-4 pairs of images are used as input to the siamese network. For each image pair we choose 3,000 positive correspondences and for each of them, one negative correspondence is randomly selected.

In inference mode (testing), the network only takes a single image as input to create the feature maps, so the runtime of inference scales linearly with the resolution of the image and also with the number of images involved. Specifically, our network infers an image at about 10 Hz on a Nvidia GPU 1080 Ti and at 2 Hz on a Nvidia GPU 750.

Chapter 5

Evaluation & Discussion

In the following chapter we evaluate our methods on the task of image matching.

5.1 Datasets and metrics

Datasets: We run the experiments on Carla Benchmark [von Stumberg et al., 2020], which includes 27,000 RGB images, the registered depth images, and the corresponding poses. The CARLA-Benchmark benchmarks 3 weather conditions with 9,000 samples for each. In each individual weather condition, 3 sequences with different image contents are included: the sequence 1 is a road with few landmarks in the roadside, the sequence 2 contains high parallax with trees in the near, while sequence 3 include a forest in the far and buildings in the near providing enough parallax. Examples on three different sequences are demonstrated in Figure 5.1. The statistics details are summarized in Table 5.1.

sequence	weather condition	scenery content	#cameras	total
episode0	WetNoon	wild road	6	3000
episode1	WetNoon	trees in near parallax	6	3000
episode2	WetNoon	far forest and near buildings	6	3000
episode3	SoftRainNoon	wild road	6	3000
episode4	SoftRainNoon	trees in near parallax	6	3000
episode5	SoftRainNoon	far forest and near buildings	6	3000
episode6	WetCloudySunset	wild road	6	3000
episode7	WetCloudySunset	trees in near parallax	6	3000
episode8	WetCloudySunset	far forest and near buildings	6	3000
total	3	-	-	27000

Table 5.1: Statistics of the sequences at the CARLA-Benchmark. It describes the weather scenarios under which data was collected, sequences contents, the number of cameras, and the images.

Figure 5.1: An overview of the Carla Benchmark dataset [von Stumberg et al., 2020]. Each row presents a different point of view; each image pair shows the comparison of the same place in the same sequence in different weather.

Performance metrics: Mikolajczyk et al. [Mikolajczyk and Schmid, 2005, Miksik and Mikolajczyk, 2012] propose to use metrics of *recall*, *repeatability*, *1-precision* to describe the characteristics of a feature's performance, and are widely used as standard measures. The definition of the three terms are:

- *Repeatability* is the ration between the number of correspondences and the the total number of matches:

$$repeatability = \frac{\#correspondences}{\#correct\ matches + \#false\ matches}$$

- *Recall* is the number of correctly matched regions with respect to the number of corresponding regions between two images of the same scene:

$$recall = \frac{\#corrected\ matches}{\#correspondences}$$

- *1-Precision* is the number of false matches relative to the total number of matches:

$$1 - precision = \frac{\#false\ matches}{\#correct\ matches + \#false\ matches}$$

However, there are some subtleties to a feature's performance that are missed by only using these measures. Other than that, the evaluated objects are patches, and the matches criteria are different from our tasks. Inspired by binary descriptor by Heinly [Heinly et al., 2012], we use the metrics of *putative match ratio*, *precision* and *matching score*, *recall* to evaluate our proposed methods.

- *Putative Match Ratio* quantifies the selectivity of the descriptor in terms of the fraction of the detected features initially identified as a match, and defined as following:

$$putative\ match\ ratio = \frac{\#putative\ matches}{\#features}$$

  This metric is directly influenced by the matching criteria: a less restrictive matching criteria generate a higher putative match ratio, whereas a criteria that is too restrictive will discard potentially valid matches and will decrease the putative match ratio.

- *Precision* defines the inlier ratio of the putative matches, as determined by geometric verification:

$$precision = \frac{\#correct\ matches}{\#putative\ matches}$$

It is also influenced by many of the same factors that influenced the putative match ratio, but the consequences are different. For instance, while a less restrictive matching criteria will increase the putative match ratio, it will decrease the precision as a higher number of incorrect matches will be generated.

- *Matching Score* defines the number of initial features that will result in inlier matches:

$$matching\ score = \frac{\#correct\ matches}{\#features}$$

Similar to the previous two metrics, the matching score varies from the matching criteria.

- *Recall* describes the number of identified ground truth matches:

$$recall = \frac{\#correct\ matches}{\#corrrespondences}$$

The correspondences are the matches given only the keypoint location in both images and are matched beforehand serving as ground truth matches. A low recall could mean the matching criterion is too strict or the data is too complex.

The relationship of the different matches, features and correspondences are shown in Figure 5.2.

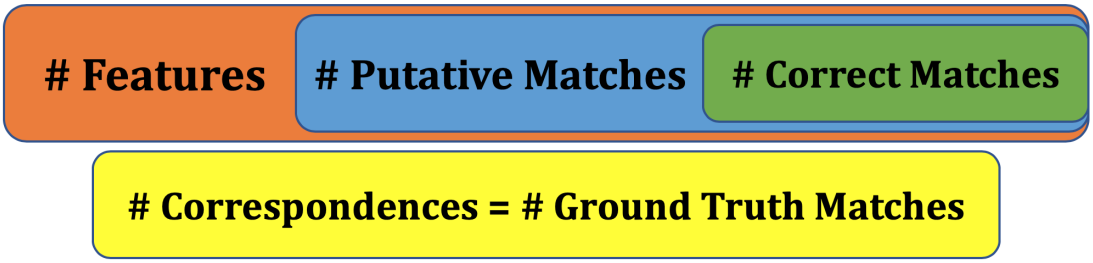


Figure 5.2: The relationship of features, putative matches, correct matches, and correspondences.

In other works [Balntas et al., 2017, Luo et al., 2020], the terminology of Keypoint repeatability (*%Rep.*), Mean matching accuracy (*%MMA*), Matching score (*%M.S.*), and *%Recall* is used. However, their definitions are exactly the same. In the following part, we use the notation in HPatches, namely *putative match ratio*, *mean matching accuracy* and *matching score* and *recall*.

Match Criteria: To compute putative matches, nearest neighbor brute searching is performed followed by a cross check and a ratio style test [Lowe, 2004]. The cross check aims to select the pairs (i, j) such that for i-th query descriptor the j-th descriptor in the matcher's collection is the nearest and vice versa. The ratio rest is that, for each descriptor from the reference image, the two nearest neighbors descriptors in the query image are found as the best and the second best matches. If the ratio of the distance between the referred descriptor and the best match to the distance between the referred descriptor and the second best match is less than a certain threshold, then the best match is selected as a putative match, otherwise both matches are rejected. According to [Lowe, 2004], the ratio is selected as 0.8.

To determine if a putative match is correct, we use ground truth geometric information to wrap the keypoints from the referred image to the query images, which has been described in Section 4.2.1. The matched points that are within a error threshold k pixels of each other in shared view are considered as inlier matches.

To avoid abuse of notation, we denote the different metrics for different threshold k as *%PTT*, *MMA@k* and *MS@k* and *RC@k putative match ratio*, *mean matching accuracy* and *matching score* and *recall* correspondingly.

5.2 Ablation study

We conduct ablation studies on the Carla Benchmark Dataset in three aspects, namely, the comparison to original loss function: original contrastive loss, triplet loss; different sampling strategy; and the choice of up-sampling layer. In all ablation studies, we use edge points detected by Canny Edge Detector [Canny, 1986], the lower threshold is set to 50, and the ratio between the higher threshold and lower threshold is set to 3 as suggested by Canny. To demonstrate the influence of different losses, we show *MMA* metric for the error threshold of 5 [Liu et al., 2019, DeTone et al., 2018, Luo et al., 2020], which means that the matches are considered right if the distance of the matched keypoint is less than 5 in shared view.

Comparison of different loss: We consider three different losses, namely, contrastive loss in Equation (4.4), triplet loss in Equation (4.5), and our new loss but without the positive margin in Equation (4.15). Table 5.2 summarizes results of the proposed method and other loss functions for error threshold of 5. The proposed loss function achieves the best performance compared to other losses.

Negative sampling strategies: To demonstrate the benefit of our proposed sampling method, we test on two commonly-used sampling methods, namely uniform sampling(without any sampling strategies) and hard negative mining [Choy et al., 2016, Florence et al., 2018]. For all different sampling strategies, we use trained with

	contrastive	triplet	w/o Pos Margin	proposed
Sequence 1	78.78	74.52	81.56	**82.17**
Sequence 2	80.78	81.45	82.02	**82.17**
Sequence 3	75.26	77.65	80.87	**81.27**

Table 5.2: Comparison of different losses on three different sequences for metric *MMA@5*.

our proposed new loss and use the up-sampling layer with pixel-shuffle [Shi et al., 2016]. The result for this ablation study is demonstrated in Table 5.3. As shown in the table, our proposed sampling methods outperforms uniform sampling by large margin, e.g. over 10%(13%) on sequence 1. However, on sequence 2, hard negative mining strategy obtain a higher *MMA*. Considering the scenery in sequence 2, which include lots of trees in the near, providing lots of fine textures, this is because hard negative mining could find out finer structure in the images.

	uniform	hard negative	proposed
Sequence 1	69.17	81.25	**82.17**
Sequence 2	73.34	**83.19**	82.17
Sequence 3	71.56	80.52	**81.27**

Table 5.3: Comparison of different sampling methods on different sequences for metrics *MMA@5*.

	transposed	bilinear	proposed
Sequence 1	80.76	81.15	**82.17**
Sequence 2	81.15	82.01	**82.17**
Sequence 3	80.56	80.05	**81.27**

Table 5.4: Comparison of different up-sampling layers on different sequences for metric *MMA@5*.

Up-sampling method: To illustrate the advance of pixel-shuffle up-sampling method, we test on different up-sampling layers, namely Transposed Convolution or Deconvolution and Bilinear Interpolation. The only modification for this ablation study is that we replace all the up-sampling layers during the up-sampling path of the U-Net. The result is summarized in Table 5.4.

5.3 Comparison with other methods

We compare our method with different classical features pipelines: SIFT [Lowe, 2004], SURF [Bay et al., 2006], ORB [Rublee et al., 2011], and A-KAZA [Alcan-

tarilla, 2011], and two state-of-the-art deep learning methods: Superpoint [DeTone et al., 2018] and GIFT [Liu et al., 2019], using the authors' released trained model for deep learned features and OpenCV for the classical features. Superpoint [DeTone et al., 2018] localizes keypoints and interpolates descriptors of these keypoints directly on a feature map of a vanilla CNN. GIFT [Liu et al., 2019] treat transformed versions of an image as groups of transformation and extract features on these groups.

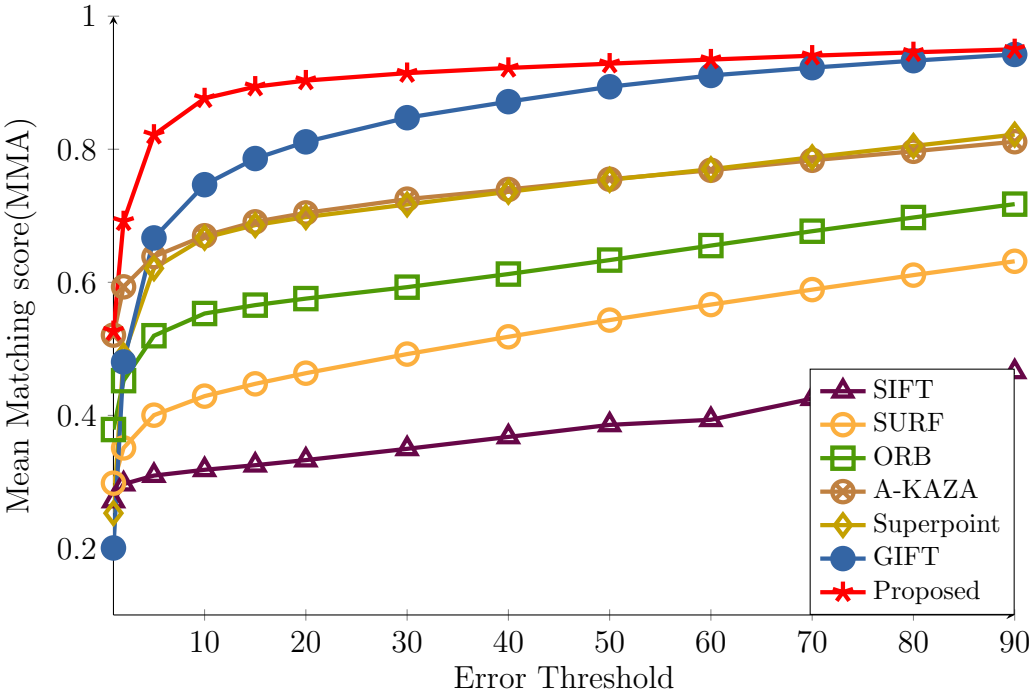


Figure 5.3: *MMA* at a different error threshold, experiment running on sequence 1.

Since our task is not aiming for feature matching, but for benefit of direct SLAM method, we do not use the keypoints generated in consideration of scale or rotation, but for high gradient points in images. Nevertheless, if we detect high gradient points in the image as keypoints, there are too many keypoints(50,000 - 200,000) for feature matching, which is not realistic for evaluation. Instead, we use Canny Edges Detector [Canny, 1986]. By tuning the lower threshold and the higher threshold, we finally set the lower threshold as 50 and the higher threshold as 150, which results in 5,000 - 12,000 keypoints for one image in the final place.

For fair comparison, we use the same keypoints (Canny Edges) as the feature detectors for all deep learned methods during evaluation. As for classical feature methods, since their own detector is implemented in OpenCV, we use their own detectors. Results are summarized in Table 5.5. The result for *MMK* under different error threshold on 3 different sequences is shown in Figure 5.3, Figure 5.4, and

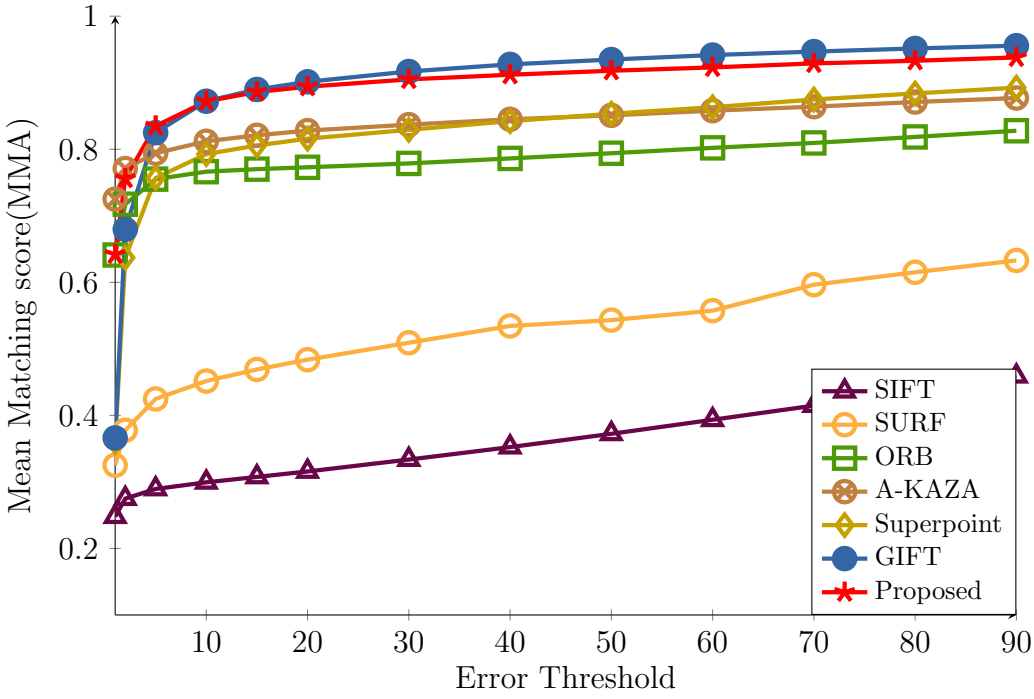


Figure 5.4: *MMA* at a different error threshold, experiment running on sequence 2.

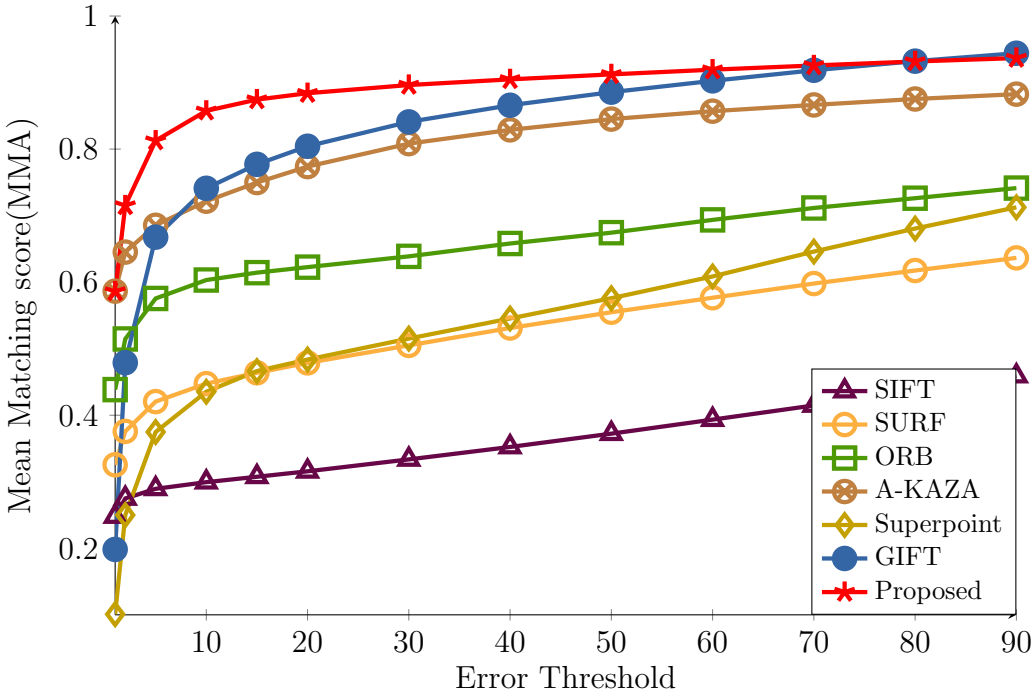


Figure 5.5: *MMA* at a different error threshold, experiment running on sequence 3.

detector	Own detector				Canny Edges Detector		
descriptor / sequences	SIFT	SURF	ORB	A-KAZA	GIFT	Superpoint	Ours
Seq 1	28.94	40.03	51.97	63.87	66.67	19.48	**82.17**
Seq 2	30.95	42.47	75.51	**89.37**	82.50	75.67	83.37
Seq 3	30.26	42.05	57.55	68.55	66.77	37.48	**81.27**
average	30.05	41.25	61.68	73.93	71.98	44.21	**82.27**

Table 5.5: Comparison with *%MMA@5* of our proposed method, classical feature detection and extraction method and the state-of-the-art methods on three different sequences of the Carla benchmark dataset [von Stumberg et al., 2020].

detector	Own detector				Canny Edges Detector		
descriptor / sequences	SIFT	SURF	ORB	A-KAZA	GIFT	Superpoint	Ours
Seq 1	9.67	12.46	18.66	**27.25**	6.10	6.90	*20.84*
Seq 2	10.47	13.45	35.72	**54.47**	12.70	10.21	*20.84*
Seq 3	10.47	13.12	19.93	**29.10**	5.96	2.71	*20.75*

Table 5.6: Comparison with *%MS@5* of our proposed method, classical feature detection and extraction method and the state-of-the-art methods on three different sequences of the Carla benchmark dataset [von Stumberg et al., 2020].

detector	Own detector				Canny Edges Detector		
descriptor / sequences	SIFT	SURF	ORB	A-KAZA	GIFT	Superpoint	Ours
Seq 1	32.84	30.21	34.35	**41.99**	8.67	7.67	*25.15*
Seq 2	33.56	31.51	47.07	**60.69**	14.91	12.9	*25.15*
Seq 3	33.32	30.89	33.94	**42.02**	8.61	6.83	*25.33*

Table 5.7: Comparison with the *PTT Ratio* of our proposed method, the classical feature detection and extraction method, and the state-of-the-art methods on three different sequences of the Carla benchmark dataset [von Stumberg et al., 2020].

Figure 5.5.

5.4 Discussion

Note that the *Putative Matches Ratio* ($\%PTT$) is affected due to the strict matching criteria, by conducting cross check and ratio test. Another reason is because of highly similar descriptors (Canny Edges), which results in a low *MS*.

As shown in Table 5.5, our proposed method outperforms all other state-of-art methods and classical methods, except on sequence 2, in which A-KAZA [Alcantarilla, 2011] performs best. The reason results from two possible perspectives: Firstly, A-KAZA uses its own feature detectors, which result in a very high *putative match ratio*, as shown in Table 5.7. However, since we are mainly focusing on improving the front-end of the direct method of SLAM, we do not focus on the putative match ratio but on the real matching numbers from high gradient regions (which are used for tracking in direct SLAM [Engel et al., 2016]). Here we use Canny edges, which have high similarity in the RGB domain. However, compared with two other state-of-art deep learned features, our method achieves a higher PTT. Other than that, another reason is that sequence 2 presents a scenery containing rich textures. There are lots of trees on the roadside creating rich textures, where ORB [Rublee et al., 2011] and A-KAZA [Alcantarilla, 2011], and two deep learned features GIFT [Liu et al., 2019] and Superpoint [DeTone et al., 2018] perform better than in the remaining two sequences. It is worth noting that the difference in texture does not affect our proposed method, where for three sequences containing different texture we perform almost the same, which we note in *Italic*. The reason is that we only consider high-gradient points (including Canny Edges), and edges do not rely much on the texture. This advantage benefits direct SLAM systems like DSO [Engel et al., 2016] and LSD-SLAM [Engel et al., 2014].

Despite a slightly worse performance on sequence 2, our proposed method achieves a better average performance on the CARLA-Benchmark overall [von Stumberg et al., 2020]. More qualitative matching result are demonstrated in Figure 5.6. A detailed comparison with *MMA* at different thresholds is illustrated in Figure 5.3 for sequence 1, Figure 5.4 for sequence 2, and Figure 5.5 for sequence 3. It can be seen that on sequence 1 and sequence 3 our method outperforms the other methods except on sequence 3, when the error threshold is extremely large (@90). For sequence 2, our proposed method performs better at low error threshold (@5-15) but then worse than GIFT [Liu et al., 2019].

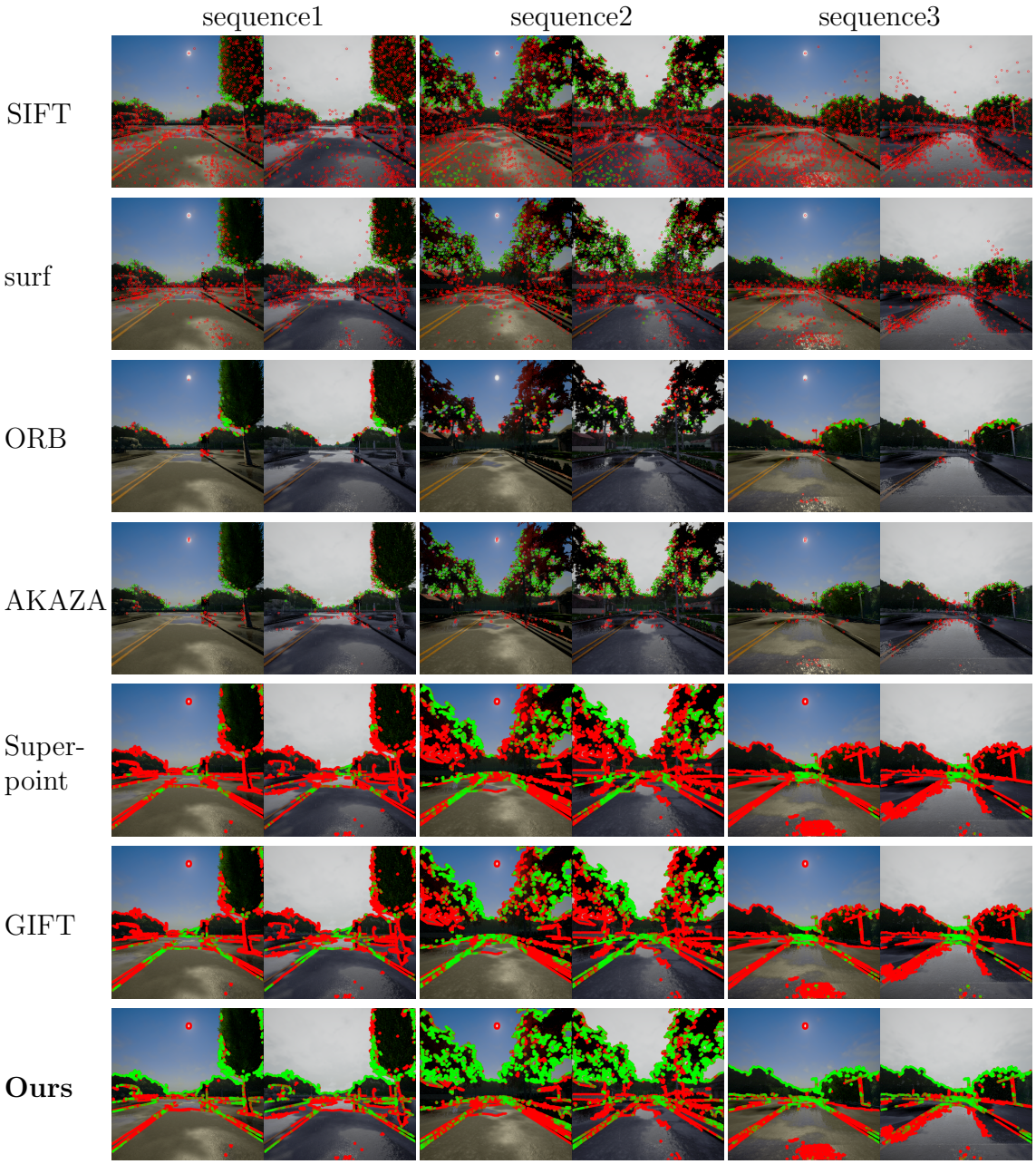


Figure 5.6: Matching results(Error threshold @5) on 3 sequences for different methods, the red circles represent all the keypoints, while green points represent the matched inliers. It can be seen that our method produce a much denser matched Canny Edges than traditional methods and the state-of-art deep learning methods.

Chapter 6

Conclusion

6.1 Discussion

Deep learning-based approaches excel in robustness when interpreting visual images of the scene, hence achieving invariance with respect to the illumination conditions that pose the biggest challenge to direct SLAM methods. In this work we propose to replace the original input sensor data (RGB images) with learned feature maps, which provide improved pose estimation accuracy and robustness compared with the baseline method using only the intensity (grayscale) channel. We also demonstrate that the sampling of correspondences is a central aspect when constructing the losses for learning dense feature maps; we focus on the distribution of the high-dimensional sample related to their distances. Finally, we leverage the advantages in the definition of the contrastive loss and the triplet loss in order to give the output feature maps more constraints to generate more consistent learned feature maps.

It is worth noting that the proposed algorithm is orthogonal to several existing direct and feature-based SLAM approaches by embedding the process of learning features in the feature matching process part.

6.2 Outlook

This learned SLAM approach still suffers from its increased computational cost, which means that it remains a challenge to run it on real robots on the fly. One possible future improvements would be to plant the neural network with a pretrained model onto a chip board.

Secondly, in oder to extract more information around high-gradient pixels, the neighborhood of the pixels could be used to learn a more informative feature map.

List of Figures

Acronyms and Notations

HRC Human-Robot Collaboration

HRI Human-Robot Interaction

HRT Human-Robot Team

MMA Mean Matching Accuracy

MC Macthing Score

RC Recall

PTT Ratio Putative Matches Ratio

HDFM High-Dimensional Feature Maps

PCA Principle Component Analysis

FCN fully convolutional neural network

Bibliography

[Aitken et al., 2017] Aitken, A., Ledig, C., Theis, L., Caballero, J., Wang, Z., and Shi, W. (2017). Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*.

[Alcantarilla, 2011] Alcantarilla, P. F. (2011). Fast explicit diffusion for accelerated features in nonlinear scale spaces.

[Balntas et al., 2017] Balntas, V., Lenc, K., Vedaldi, A., and Mikolajczyk, K. (2017). Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5173–5182.

[Balntas et al., 2016] Balntas, V., Riba, E., Ponsa, D., and Mikolajczyk, K. (2016). Learning local feature descriptors with triplets and shallow convolutional neural networks.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

[Bromley et al., 1994] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

[Chen et al., 2017] Chen, W., Chen, X., Zhang, J., and Huang, K. (2017). Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412.

[Choy et al., 2016] Choy, C. B., Gwak, J., Savarese, S., and Chandraker, M. (2016). Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052—1067.

[DeTone et al., 2018] DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236.

[Engel et al., 2016] Engel, J., Koltun, V., and Cremers, D. (2016). Direct sparse odometry. In *arXiv:1607.02565*.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *eccv*.

[Florence et al., 2018] Florence, P. R., Manuelli, L., and Tedrake, R. (2018). Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*.

[Gálvez-López and Tardós, 2012] Gálvez-López, D. and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[Godard et al., 2017] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279.

[Gomez-Ojeda et al., 2018] Gomez-Ojeda, R., Zhang, Z., Gonzalez-Jimenez, J., and Scaramuzza, D. (2018). Learning-based image enhancement for visual odometry in challenging hdr environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 805–811. IEEE.

[Handa et al., 2014] Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531. IEEE.

[Hao et al., 2018] Hao, J., Dong, J., Wang, W., and Tan, T. (2018). Deepfirearm: Learning discriminative feature representation for fine-grained firearm retrieval. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3335–3340. IEEE.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[Heinly et al., 2012] Heinly, J., Dunn, E., and Frahm, J.-M. (2012). Comparative evaluation of binary features. In *European Conference on Computer Vision*, pages 759–773. Springer.

[Hermans et al., 2017] Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

[Hoffer and Ailon, 2015] Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

[Hughes et al., 2018] Hughes, L. H., Schmitt, M., and Zhu, X. X. (2018). Mining hard negative samples for sar-optical image matching using generative adversarial networks. *Remote Sensing*, 10(10):1552.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[Keller et al., 2018] Keller, M., Chen, Z., Maffra, F., Schmuck, P., and Chli, M. (2018). Learning deep descriptors with scale-aware triplet networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2762–2770.

[Kendall et al., 2015] Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

[Kerl et al., 2013] Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. In *icra*.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan.

[Lehnen and Wesenberg,] Lehnen, A. and Wesenberg, G. E. The sphere game in n dimensions.

[Liang et al., 2019] Liang, L., Cao, J., Li, X., and You, J. (2019). Improvement of residual attention network for image classification. In Cui, Z., Pan, J., Zhang, S., Xiao, L., and Yang, J., editors, *Intelligence Science and Big Data Engineering. Visual Data Engineering*, pages 529–539, Cham. Springer International Publishing.

[Liu et al., 2017] Liu, M.-Y., Breuel, T., and Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708.

[Liu et al., 2019] Liu, Y., Shen, Z., Lin, Z., Peng, S., Bao, H., and Zhou, X. (2019). Gift: Learning transformation-invariant dense visual descriptors via group cnns. In *Advances in Neural Information Processing Systems*, pages 6990–7001.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Luo et al., 2019] Luo, Z., Shen, T., Zhou, L., Zhang, J., Yao, Y., Li, S., Fang, T., and Quan, L. (2019). Contextdesc: Local descriptor augmentation with cross-modality context.

[Luo et al., 2020] Luo, Z., Zhou, L., Bai, X., Chen, H., Zhang, J., Yao, Y., Li, S., Fang, T., and Quan, L. (2020). Aslfeat: Learning local features of accurate shape and localization. *arXiv preprint arXiv:2003.10071*.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630.

[Miksik and Mikolajczyk, 2012] Miksik, O. and Mikolajczyk, K. (2012). Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2681–2684. IEEE.

[Mishchuk et al., 2017] Mishchuk, A., Mishkin, D., Radenoviundefined, F., and Matas, J. (2017). Working hard to know your neighbor's margins: Local descriptor learning loss. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4829–4840, Red Hook, NY, USA. Curran Associates Inc.

[Mou et al., 2017] Mou, L., Schmitt, M., Wang, Y., and Zhu, X. X. (2017). A cnn for the identification of corresponding patches in sar and optical imagery of urban scenes. In *2017 Joint Urban Remote Sensing Event (JURSE)*, pages 1–4. IEEE.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Mur-Artal and Tardos, 2017] Mur-Artal, R. and Tardos, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[Newcombe et al., 2011] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *Proceedings of the 2011 International Conference on Computer Vision*, page 2320–2327, USA. IEEE Computer Society.

[Odena et al., 2016] Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*.

[Piasco et al., 2018] Piasco, N., Sidibé, D., Demonceaux, C., and Gouet-Brunet, V. (2018). A survey on visual-based localization: On the benefit of heterogeneous data. *Pattern Recognition*, 74:90–109.

[Piasco et al., 2019] Piasco, N., Sidibé, D., Gouet-Brunet, V., and Demonceaux, C. (2019). Learning scene geometry for visual localization in challenging conditions. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9094–9100.

[Porav et al., 2018] Porav, H., Maddern, W., and Newman, P. (2018). Adversarial training for adverse conditions: Robust metric localisation using appearance transfer. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1011–1018. IEEE.

[Pumarola et al., 2017] Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F. (2017). Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

[Schmidt et al., 2016] Schmidt, T., Newcombe, R., and Fox, D. (2016). Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[Shi et al., 2016] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 1874–1883.

[Simo-Serra et al., 2015] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., and Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126.

[Stühmer et al., 2010] Stühmer, J., Gumhold, S., and Cremers, D. (2010). Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium*, pages 11–20. Springer.

[Sun et al., 2014] Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996.

[von Stumberg et al., 2020] von Stumberg, L., Wenzel, P., Khan, Q., and Cremers, D. (2020). Gn-net: The gauss-newton loss for multi-weather relocalization. *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, 5(2):890–897.

[Wang et al., 2014] Wang, J., song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking.

[Wohlhart and Lepetit, 2015] Wohlhart, P. and Lepetit, V. (2015). Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3109–3118.

[Wu et al., 2017] Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848.

[Yang et al., 2018] Yang, N., Wang, R., Stuckler, J., and Cremers, D. (2018). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833.

[Zagoruyko and Komodakis, 2015] Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361.